Universidad Michoacana de San
Nicolás de Hidalgo

Faculty of Electrical Engineering

Translating Thought: Advancing First-Order Logic Interpretation with Llama2 and Z3 Solver

T H E S I S

Submitted to obtain the degree of master of science in electrical engineering

By

Ángel Misael Pelayo Gómez

Supervisor: Dr. Jaime Cerda Jacobo

Morelia, Michoacán, México
February, 2025

TRANSLATING THOUGHT: IMPROVING FIRST-ORDER LOGIC INTERPRETATION WITH LLAMA 2 AND Z3 SOLVER

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de **Ángel Misael Pelayo Gómez.**

Dr. Luis Eduardo Gamboa Guzmán
Presidente del Jurado

Dr. Jaime Cerda Jacobo
Director de Tesis

Dr. José Antonio Camarena Ibarrola
Vocal

Dr. José Ortíz Béjar
Vocal

Dr. Mario Graff Guerrero
Revisor Externo (INFOTEC)

Dr. J. Aurelio Medina Rios
Jefe de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas)

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Junio 2024

To God above, who guides my way,
in challenges, through night and day.
Your strength and wisdom in my quest,
in pursuit of dreams, I am blessed.
To family, with love so deep,
your faith in me, I'll always keep.
In times of trial, in times of cheer,
your unwavering support, I hold dear.
To mentors, friends, who light my path,
with guidance, laughs, and sometimes wrath.
Your wisdom, like a guiding star,
has brought me near, has brought me far.

"Put your heart, mind, and soul into even your smallest acts. This is the secret of success." Swami Sivananda.

Acknowledgements

First and foremost, I extend my deepest gratitude to my supervisor, Dr. Jaime Cerda, for his invaluable guidance, support, and expertise throughout this research journey. His insights and suggestions have been instrumental in shaping this thesis.

I am immensely grateful to Conacyt for their generous scholarship, which has been essential in supporting my academic pursuits and research. Their financial assistance has been a cornerstone of my success.

My sincere appreciation goes to the Universidad Michoacana de San Nicolás de Hidalgo (UMSNH) and the Faculty of Electrical Engineering for providing me with the necessary resources, including access to high-performance computing facilities, which were crucial for training my model. The availability of GPUs and other technical support has significantly contributed to the accomplishment of this research.

I would like to acknowledge and thank all my professors during my time at UMSNH for their invaluable instruction and mentorship. Special thanks to Dr. Felix Calderon, Dr. Antonio Camarena, Dr. Jaime Cerda, Dr. Jose Ortiz, Dr. Luis Gamboa, and Dr. Claudio Fuerte for their teachings and wisdom, which have profoundly influenced my academic and professional growth.

A heartfelt thanks goes to my classmates and friends, Victor, Jaime, Brenda, Jehosafat, and my brother Diego, for their companionship, support, and invaluable contributions to my academic journey. Our shared experiences and discussions have been a source of motivation and inspiration.

Lastly, I must express my profound gratitude to my family and friends for their unwavering support and encouragement throughout my academic journey. This accomplishment would not have been possible without their belief in me and constant backing.

Angel Misael Pelayo Gomez

"Gratitude is the memory of the heart." Lao Tsé.

Translating Thought: Advancing First-Order Logic Interpretation with Llama2 and Z3 Solver

Ángel Misael Pelayo Gómez

Abstract

Keywords: Natural Language Processing, First-Order Logic, Large Language Models, Llama2, Automated Reasoning, Z3 Theorem Prover, Neural Network Fine-Tuning, Ludwig, Machine Learning, AI and Formal Logic Integration, NL to FOL Translation, FOL to SMT-Lib parser.

Our thesis tackles the challenge of translating natural language (NL) into first-order logic (FOL), an area where traditional methods have struggled due to the inherent ambiguity of natural language. By addressing this ambiguity, we aim to enhance the clarity and precision of automated reasoning. Our study focuses on utilizing the latest 7 billion parameter version of Llama2, optimized with techniques like LoRa and quantization, and integrating it with Z3, a state-of-the-art theorem prover. This integration facilitates the automatic determination of the truth value of a set of premises and a conclusion, ensuring logical consistency and accuracy. We employ a dual approach that involves fine-tuning Llama2 for enhanced NL to FOL translation and leveraging Z3 to check the satisfiability of these premises.

We are impressed by the notable results of our study, with the model achieving nearly 70 percent accuracy in FOL validity for a set of premises and a conclusion. This performance not only aligns with but also slightly surpasses current benchmarks in the field, indicating a high rate of correct translations. The integration of a generative model like Llama2 with a deterministic solver like Z3 represents a significant advancement in the realm of automated reasoning and natural language processing. Our work contributes to the field by presenting a novel method that offers more accurate and logically consistent NL to FOL translations than existing approaches. It opens new pathways for sophisticated, reliable AI systems, bridging the gap between machine learning and formal logic, and enhancing the ability to automatically evaluate logical validity.

"The ability to present an idea is as important as the idea itself." Aristotle.

Translating Thought: Advancing First-Order Logic Interpretation with Llama2 and Z3 Solver

Ángel Misael Pelayo Gómez

Resumen

Palabras clave: Procesamiento del Lenguaje Natural, Lógica de Primer Orden, Modelos de Lenguaje Grandes, Llama2, Razonamiento Automatizado, Z3 Demostrador de Teoremas, Ajuste Fino de Redes Neuronales, Ludwig, Aprendizaje Automático, Integración de IA y Lógica Formal, Traducción de NL a FOL, FOL a parser SMT-Lib.

Nuestra tesis aborda el desafío de traducir el lenguaje natural (NL) a lógica de primer orden (FOL), un área donde los métodos tradicionales han tenido dificultades debido a la ambigüedad inherente del lenguaje natural. Al abordar esta ambigüedad, buscamos mejorar la claridad y precisión del razonamiento automatizado. Nuestro estudio se centra en utilizar la última versión de 7 mil millones de parámetros de Llama2, optimizada con técnicas como LoRa y cuantización, e integrarla con Z3, un demostrador de teoremas de vanguardia. Esta integración facilita la determinación automática del valor de verdad de un conjunto de premisas y una conclusión, asegurando consistencia lógica y precisión. Empleamos un enfoque dual que involucra el ajuste fino de Llama2 para una mejor traducción de NL a FOL y el uso de Z3 para verificar la satisfacibilidad de estas premisas.

Estamos impresionados por los resultados notables de nuestro estudio, con el modelo logrando casi un 70 por ciento de precisión en la validez de FOL para un conjunto de premisas y una conclusión. Este rendimiento no solo se alinea con, sino que también supera ligeramente los puntos de referencia actuales en el campo, indicando una alta tasa de traducciones correctas. La integración de un modelo generativo como Llama2 con un solucionador determinista como Z3 representa un avance significativo en el ámbito del razonamiento automatizado y el procesamiento del lenguaje natural. Nuestro trabajo contribuye al campo presentando un método novedoso que ofrece traducciones de NL a FOL más precisas y lógicamente consistentes que los enfoques existentes. Abre nuevos caminos para sistemas de IA sofisticados y confiables, cerrando la brecha entre el aprendizaje automático y la lógica formal, y mejorando la capacidad de evaluar automáticamente la validez lógica.

"La capacidad de presentar una idea es tan importante como la idea misma." Aristóteles.

Contents

List of Figures

List of Tables

Introduction

The task of translating natural language (NL) sentences into logical expressions poses a significant challenge, even for humans. University students, when confronted with this task, have demonstrated difficulties in achieving a satisfactory level of performance (Barker-Plummer et al., 2000). The complexity of this translation arises from the inherent ambiguity of natural language and the numerous ways a single idea can be conveyed using different phrasings.

The intricacy of the translation process becomes apparent due to the absence of a fixed set of rules. Natural language is replete with ambiguities, and there are multiple ways to express the same idea using different words and structures. For instance, consider the sentence: "If a student doesn't finish his homework, he will fail the exam." This sentence must be translated into first-order logic (FOL) as "$\forall x(Student(x) \land \neg DoHomework(x)) \rightarrow FailExam(x)$)". However, the inherent flexibility of natural language allows for alternative phrasings such as "The failure to complete homework by any student results in an exam failure" or "Unless a student completes his homework, he will not succeed in the exam", conveying the same meaning.

Hence, the task of translating natural language propositions to first-order logic presents formidable challenges that elude classical approaches, often implemented as algorithmic programs. While we found a project (discussed in section 3.1) that represents a commendable effort in rule-based translation algorithms, this approach encounter inherent limitations. The system exhibit competent performance with simpler sentence structures but struggle significantly when dealing with complex, real-world texts that are typical in natural language.

In contrast, the application of machine learning paradigms emerges as the ideal path forward for addressing these non-specific translation tasks. By harnessing the capabilities of machine learning, particularly through the utilization of Large language models (LLMs) based on transformer neural networks such as **Llama2** (Touvron et al., 2023) (see section 1.4 for more details), we aim to overcome the limitations that traditional algorithmic methods encounter in capturing the intricacies and subtleties inherent in natural language, converting NL sentences into FOL formulas, commonly known as logic formalization of natural language sentences. (Perikos et al., 2023)

While the literature presents various methods for parsing NL into FOL, including techniques utilizing sequential models like LSTM and GRU (Singh et al., 2020), as well as models incorporating the concept of attention (Dong & Lapata, 2016), these approaches contribute to a diverse landscape in addressing the challenges of natural language translation into FOL. Notably, Yang et al. (Yang et al., 2023) have also explored a similar pathway, utilizing the power of the first version of Llama for NL to FOL translation. Our

"Everything is theoretically impossible until it is done." Robert A. Heinlein.

thesis extends this evolving paradigm by employing a unique approach with the latest version of Llama, fine-tuned using Ludwig (see Appendix A), and introduces the concept of automated reasoning derived from natural language sentences. While there are existing works that employ machine learning for reasoning (Korovin, 2023; Han et al., 2022b), they often apply these techniques directly to reasoning tasks. In contrast, this thesis presents a novel approach that synergistically combines the interpretive power of LLMs with the precision and determinism of solvers like Z3, a state-of-the-art theorem prover from Microsoft Research (see Appendix B). This fusion allows for an unprecedented level of accuracy and depth in reasoning tasks, leveraging the strengths of both machine learning and traditional logic solvers.

Our methodology, detailed in chapter 4, illustrates how we leverage the Large Language Model (LLM) to translate natural language into structured first-order logic representations, and then utilize Z3 (de Moura & Bjørner, 2008) to engage in sophisticated automated reasoning with these translations. This not only bridges the nuanced and often ambiguous nature of natural language with the logical rigidity of FOL but also enhances the capabilities of machines in understanding and interacting with human language.

To the best of our knowledge, no existing work has ventured into this territory, making our research the pioneering effort in, we believe our contributions extend beyond the development of a state-of-the-art model; they mark a significant step forward in the intersection of Natural Language Processing and formal logic.

I.1 Research Justification and Significance

Our thesis, addresses a critical gap in this field of automated reasoning. The justification for undertaking this research stems from several compelling considerations that underscore the significance and timeliness of this work.

Bridging Linguistic Ambiguity and Logical Precision

Natural language's inherent ambiguity presents a formidable barrier to precise logical interpretation, a necessity for advanced reasoning tasks. Existing methods, while making substantial contributions, often fall short in handling the complexity and diversity of natural language expressions. This research introduces a novel methodology that leverages the latest advancements in machine learning, embodied by Llama2, and the logical rigor of Z3 Theorem Prover, promising a more sophisticated approach to resolving this enduring challenge.

Filling the Research Gap

Despite the advances in both NLP and automated reasoning, the intersection where these disciplines converge—translating natural language into first-order logic (FOL) with high accuracy and reliability—remains underexplored. Our thesis ventures into relatively uncharted territory, aiming to establish a new benchmark for accuracy and efficiency in this domain. The integration of a state-of-the-art language model with a theorem prover represents an innovative leap forward, potentially setting a new standard for future research.

"AI is neither good nor evil. It's a tool. It's a technology for us to use." Oren Etzioni.

Promoting Interdisciplinary Research

This thesis embodies the intersection of linguistics, computer science, and logic, promoting a cross-disciplinary approach that enriches our understanding of language and thought. By drawing upon the strengths of each field, this research not only tackles a specific technical challenge but also encourages further interdisciplinary collaboration, potentially yielding insights that extend far beyond the immediate scope of this work.

I.2 General Objective

The primary aim of our work is to create an innovative method for translating English natural language into First-Order Logic (FOL) expressions. We achieve this by employing the 7 billion parameter version of Llama2, fine-tuned with Ludwig. Once translated into FOL, we use Z3 to validate the formulas, determining the truth value of a set of premises and their conclusion. To train Llama2, we balanced the size of the model with computational efficiency, utilizing advanced techniques, specifically LoRa (Hu et al., 2021) and quantization (Dettmers et al., 2023), to optimize the model's performance while maintaining a manageable computational footprint.

FOL helps bridge the gap between the fluid, unstructured nature of natural language (NL) and the structured of The Satisfiability Modulo Theories Library (SMT-LIB) that is used by Z3 (see more details of SMT, SMT-LIB and Z3 in appendix B), enabling the validation of logical formulas with the deterministic precision of Z3. By leveraging SMT-LIB, our approach ensures that the translation of natural language to logic is both accurate and compatible with existing verification tools.

Ultimately, our work aims to establish a comprehensive process that converts natural language inputs into logically valid conclusions. This involves assessing the truth value of these conclusions within a predefined set of premises. Additionally, by integrating Z3, we are able to showcase the model's capability in validating First-Order Logic representations, ensuring that a given formula is indeed a valid FOL formula.

Through this integrative approach, the thesis aspires to contribute significantly to advancing the convergence of natural language understanding, computational logic, and artificial intelligence. Our goal is to develop a model that is not only powerful and computationally robust but also efficient and practically applicable in real-world scenarios. The envisioned end-to-end process, spanning from natural language input to definitive truth value assessment, symbolizes a significant leap forward in the capability of AI systems to engage in sophisticated logical reasoning and validation.

I.3 Specific Objectives

1. **Understand Large Language Models (LLMs):** Explore the mechanics of neural networks and LLMs to understand their functionality and harness their power for complex natural language processing tasks.

2. **Comprehend the NL to FOL Translation Process:** Gain a deep understanding of first-order logic (FOL) and the process of translating natural language (NL) into FOL, including the principles of FOL validity.

"Effort and courage are not enough without purpose and direction." John F. Kennedy.

3. **Build a Specialized Dataset:** Design and construct a dataset tailored to facilitate the model's learning process for effective NL to FOL translation.

4. **Fine-Tune LLama2 with the Dataset:** Utilize the custom dataset to fine-tune LLama2, experimenting with various configurations through Ludwig to optimize translation performance.

5. **Develop a FOL to SMT-LIB Parser and Reasoning Process:** Create a parser for converting FOL into SMT-LIB format and establish a process for translating NL premises and conclusions to validity statements using Z3.

6. **Evaluate the Translation Process:** Assess the translation process's performance using measures that determine the precision and accuracy of NL to FOL translations.

7. **Identify Improvement Opportunities:** Analyze the process to pinpoint potential gaps and propose areas for future improvements and research in the field.

I.4 Chapter Summaries

In Chapter 1, we delve into the foundational aspects of neural networks, focusing specifically on their application in natural language processing. This chapter covers the evolution of text generation and language modeling, with an emphasis on the transformer architecture. We further explore Large Language Models, particularly LLama, and discuss advanced fine-tuning techniques such as LoRa and quantization.

In Chapter 2, we transition into the realm of formal logic, providing an introduction to First-Order Logic (FOL). This chapter covers the syntax of FOL, including symbols, terms, and formulas, as well as semantics, logical equivalence, inference rules, and FOL validity. We conclude with an exploration of the challenges in translating natural language into FOL.

In Chapter 3, we explore various systems and approaches in translating natural language to FOL. This chapter begins with an examination of POS tagging and rule-based translation methods, discussing their limitations and potential. We then shift to LLM-based FOL translators and introduce the "FOLIO" and "MALLS" datasets for NL-to-FOL reasoning, discussing the overall approach, evaluation, and results of two main papers.

In Chapter 4, we explain the core project of the thesis, which involves the application of Llama2 and Z3 in natural language processing and logical reasoning. This chapter details dataset creation and utilization, custom dataset construction techniques, and curriculum learning implementation. We focus on fine-tuning Llama2 for NL-FOL translation, detailing Ludwig's adaptation. We also provide a comprehensive discussion on the reasoning process made by Z3 after parsing FOL formulas into SMT-LIB format, and we end with the model deployment using Gradio.

In Chapter 5, we present the experiments performed during the training and the results of the more significant models once fine-tuned. This chapter encompasses experiments and analysis of results, providing a concrete evaluation of the NL-to-FOL conversion process.

"Clarity of vision is the key to achieving your objectives." Tom Steyer.

In the Conclusion and Future Work chapter, we synthesize the findings and contributions of the thesis, offering conclusions and discussing potential avenues for future work. We aim to inspire continued exploration in the fields of NLP, AI, and computational logic.

"The study of error is not only in the highest degree prophylactic, but it serves as a stimulating introduction to the study of truth." Walter Lippmann.

Chapter 1

Decoding Linguistics: Neural Networks in Natural Language Understanding

1.1 Neural Networks

Neural networks are computational systems inspired by the biological neural networks that constitute animal brains. At their core, neural networks consist of layers of interconnected nodes, or "neurons," which process data inputs to produce an output. The architecture of a neural network is depicted in Figure 1.1.

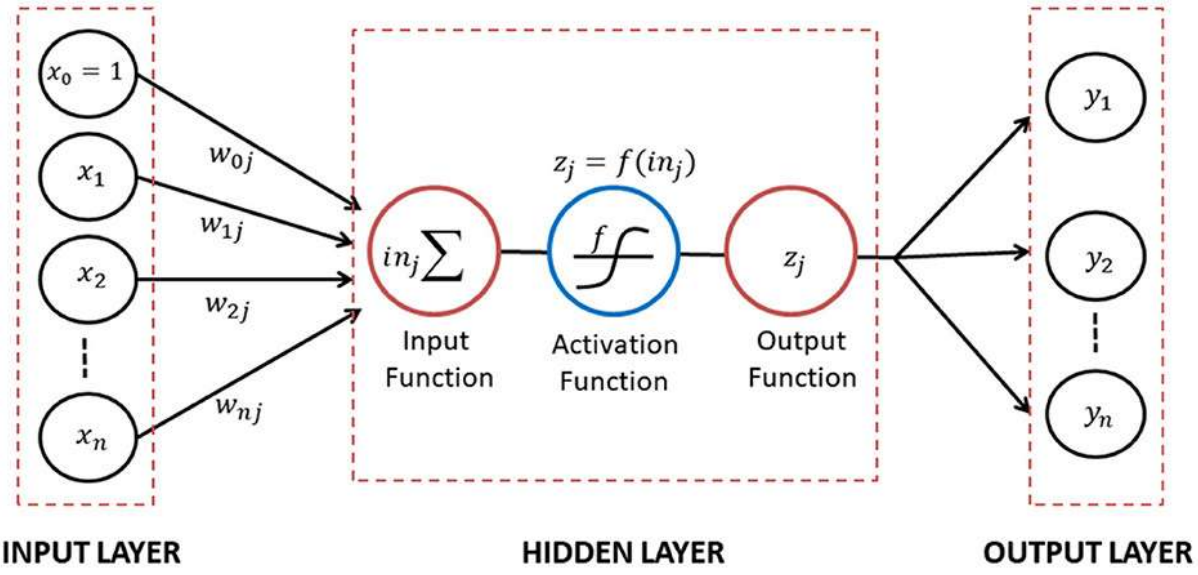

Figure 1.1: core components of a neural network's hidden layer, emphasizing the transformation of inputs through the input layer, activation function, and output layer. The activation function (f) plays a critical role in capturing non-linear patterns. Adapted from (Devianto et al., 2023)

"The neural network is this kind of technology that is not an algorithm, it is a network that has weights on it, and you can adjust the weights so that it learns. You teach it through trials. " Howard Rheingold.

Mathematically, a neural network transforms input data through layers. Each layer performs a linear transformation, typically represented by a matrix multiplication of the inputs, followed by an activation function. The general form can be expressed as:

$$output = activation(W \cdot input + b) \tag{1.1}$$

where W is the weight matrix, *input* is the input vector, b is the bias vector, and *activation* is the activation function.

1.1.1 Activation Functions

Activation functions introduce non-linear properties to the network, enabling it to learn complex data patterns. We describe some of the activation functions used in neural networks:

- **ReLU (Rectified Linear Unit):** Defined as $f(x) = max(0, x)$, it introduces non-linearity without affecting the scale of input and is computationally efficient.
- **Softmax:** It is used in the output layer of classification networks and is defined as $f(x_i) = \frac{\exp(x_i)}{\sum \exp(x_j)}$ for each input x_i, turning logits into probabilities.

1.1.2 Loss Functions

A loss function evaluates how well the neural network performs. It measures the difference between the network's predictions and the actual values. Common loss functions include:

- **Regression Loss Function:** Mean Squared Error (MSE) is commonly used, defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$, where Y_i are the actual values and $\hat{Y}_i$ are the predicted values.
- **Cross-Entropy Loss:** Used for classification tasks, it measures the performance of a classification model whose output is a probability value between 0 and 1, the mathematical definition is explained in detail in 1.2.

1.1.3 Optimization Algorithms

To "learn", a neural network must optimize its loss function. This is commonly achieved using gradient descent, an algorithm that iteratively adjusts parameters to minimize the loss function.

Gradient Descent updates the weights by calculating the gradient of the loss function with respect to the weights. This process is known as **Backpropagation**.

Adam and **AdamW** are optimization algorithms that adjust the learning rate during training. Adam combines the advantages of two other gradient descent methodologies: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).

AdamW is typically recommended over more traditional optimizers such as SGD or Adam due to its improved handling of weight decay, which is of particular importance during fine-tuning to avoid catastrophic forgetting.

"Your brain does not manufacture thoughts. Your thoughts shape neural networks. " Deepak Chopra.

1.2 The Concept of Text Generation and Language Modeling

Language modeling is a type of statistical modeling used in natural language processing (NLP) for text generation. Neural networks can generate text by learning the probability distribution of a sequence of words or characters and then sampling from this distribution to create new text sequences. A simple yet powerful approach to text generation is to train a neural network to predict the next character in a sequence, given the previous characters, known as a character-level language model.

1.2.1 Building a Bigram Character-level Language Model

Consider the process of building a bigram character-level language model, as demonstrated in the educational resource by Andrej Karpathy (Karpathy, 2021). The first step is to create a dataset for training the neural network. In this example, our dataset consists of words, and we will encode each letter of the alphabet with a unique integer token. Since we have 26 letters, and we add a special token to signify the start and end of a word, we have a total of 27 unique tokens. For a neural network, we require inputs and corresponding outputs for training for instance, in the name "evelyn", our input x would be $['.', 'e', 'v', 'e', 'l', 'y', 'n']$ and the corresponding output y would be $['e', 'v', 'e', 'l', 'y', 'n', '.']$, with the '.' symbol representing the start or end of a word.

In our bigram model, the inputs and outputs are single characters. Each character of the input is one-hot encoded into a vector of length 27, where all elements are 0 except for the position representing the character, which is 1.

The simplest form of a neural network for this task would have a single layer that transforms the one-hot encoded input vector into the probabilities of each character being the next one. We initialize the weights of this layer as a 27×27 matrix with random values. After the layer's transformation, which produces another 27-dimensional vector, we apply the softmax function to convert these values into probabilities.

We define our loss function as the negative log-likelihood, averaged over all correct characters, which in practice, is the cross-entropy loss between the predicted probabilities and the target distribution.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log(p_{i,y_i}) \tag{1.2}$$

where N is the number of characters in the training set, p_{i,y_i} is the probability assigned by the model to the true next character y_i in the context i.

With our model and loss function defined, the remaining step is to optimize the weights using backpropagation and gradient descent. This process iteratively adjusts the weights to minimize the loss function, effectively training the neural network to predict the next character in a sequence more accurately.

"In a few years artificial intelligence virtual assistants will be as common as the smart phone. " Dave Waters.

1.3 Transformer Architecture

While bigram models and their extensions can capture the immediate sequence structure, they fall short when it comes to understanding the broader context within text sequences. To address this limitation, Recurrent Neural Networks (RNNs) were introduced. RNNs process sequences by maintaining a 'memory' of previous tokens' information, which is combined with the current token's information to predict the next token in the sequence. Despite their sequential nature, RNNs struggle with parallelization and often fail to remember long-range dependencies within a sequence. A partial remedy to this was found in Long Short-Term Memory (LSTM) networks, which are better at capturing long-distance interactions.

However, the true paradigm shift occurred with the introduction of the Transformer architecture in the seminal paper "Attention Is All You Need" (Vaswani et al., 2017). Transformers sidestep the limitations of RNNs by using self-attention mechanisms (more details of attention mechanisim in section 1.3.3).

The overall architecture of the transformer model developed by vaswani et al. is characterized by an encoder-decoder structure. The encoder is responsible for mapping an input sequence of symbol representations $(x_1, \ldots, x_n)$ into a series of continuous representations $\mathbf{z} = (z_1, \ldots, z_n)$. The decoder then takes $\mathbf{z}$ and generates an output sequence $(y_1, \ldots, y_m)$ one symbol at a time. This generation process is auto-regressive, meaning that the decoder generates each symbol by considering both the continuous representations and all previously generated symbols.

The Transformer model adheres to this encoder-decoder paradigm but with a novel twist. It employs stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, which can be visualized in the respective halves of a typical Transformer architecture diagram 1.2.

1.3.1 Encoder and Decoder Stacks

Encoder: The encoder consists of a stack of $N = 6$ identical layers. Each of these layers has two sub-layers. The first sub-layer is a multi-head self-attention mechanism, allowing the layer to process different representations of the input sequence at different positions. The second sub-layer is a position-wise fully connected feed-forward network. For each of these sub-layers, a residual connection is employed, followed by layer normalization. Specifically, the output of each sub-layer is given by $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself. This design necessitates that all sub-layers and embedding layers produce outputs with the same dimension, $d_{\text{model}} = 512$, to enable the residual connections.

Decoder: The decoder mirrors the encoder with its stack of $N = 6$ identical layers, but with an additional sub-layer inserted into each layer that performs multi-head attention over the encoder stack's output. The self-attention sub-layer in the decoder is modified to prevent positions from attending to subsequent positions, ensuring causality in the generated sequence. This is achieved through masking future positions and offsetting the output embeddings by one position, ensuring that the prediction for the i-th position can only depend on the known outputs up to the $i-1$-th position. Like in the

"ChatGPT is scary good. We are not far from dangerously strong AI. " Elon Musk.

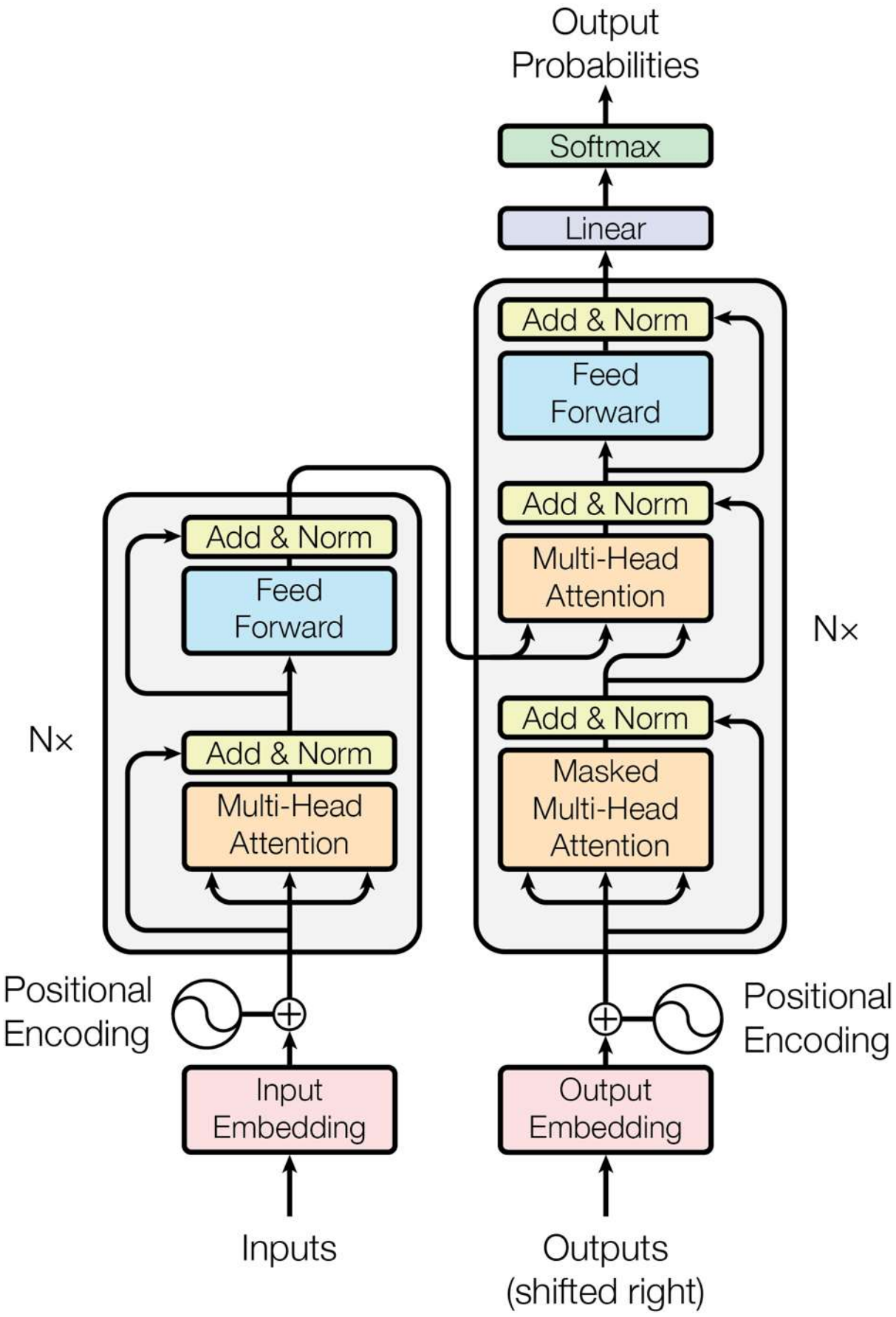

Figure 1.2: Transformer's encoder-decoder structure with stacked self-attention and point-wise, fully connected layers, transforming input sequences into outputs. Taken from (Vaswani et al., 2017).

encoder, the decoder employs residual connections followed by layer normalization for each sub-layer.

By stacking multiple layers in both the encoder and decoder, the Transformer is able to process complex input sequences and generate corresponding outputs with a high degree of parallelization and efficiency, largely due to the self-attention mechanism that eschews the sequential processing inherent in models like RNNs.

1.3.2 Positional Encoding

In contrast to RNNs, where the sequential input inherently provides position information, the Transformer architecture treats input tokens independently without any notion of their order in the sequence. To incorporate the essential information about the position of each token within the sequence, positional encodings are added to the input embeddings. This step ensures that the model can leverage the order of the tokens, which is crucial for understanding the structured information in languages.

Positional encodings are designed to be unique for each position in the sequence, allowing the model to differentiate between tokens based on their location. The Transformer adopts a specific strategy for encoding these positions, using a mix of sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$

where pos is the position of the token in the sequence, i is the dimension index, and d_{model} is the dimensionality of the token embeddings. By using this formula, each dimension of the positional encoding corresponds to a sinusoid. It allows the model to easily learn to attend by relative positions since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

The intuition behind this choice of positional encoding is that the trigonometric functions will project the position onto a high-dimensional space where the relative positions are easier to capture by the model. Furthermore, because these functions are periodic, they allow the model to generalize to sequence lengths that it has not seen during training.

This positional information is added to the token embeddings before they are input to the self-attention layers. Consequently, the model is provided with cues to determine not only the content of the tokens through the embeddings but also how the tokens are positioned in relation to one another in the sequence.

With positional encodings, the Transformer can make use of the absolute position or the relative distance between tokens when processing the sequence, a feature that is especially important for understanding language structure and generating coherent text.

1.3.3 Self-Attention

The self-attention mechanism, central to the Transformer architecture, allows the model to compute a representation of a sequence by assessing the influence of all elements in

"I visualise a time when we will be to robots what dogs are to humans, and I'm rooting for the machines. " Claude Shannon.

the sequence on each other. Unlike RNNs, which create representations from a sequence in a linear fashion, self-attention facilitates a more dynamic dependency model, enabling simultaneous consideration of the entire sequence context.

Each token in the sequence, after undergoing embedding and receiving positional encoding, is transformed into a vector, denoted as $\mathbf{x}$. The self-attention mechanism begins by projecting $\mathbf{x}$ onto three distinct vectors: Key ($\mathbf{k}$), Query ($\mathbf{q}$), and Value ($\mathbf{v}$). These vectors are derived by multiplying $\mathbf{x}$ with their respective weight matrices, which are learned during training.

The Key vector represents the aspects of a token to be compared with others, akin to the characteristics one would seek in a conversation partner. The Query vector, conversely, is representative of the token's own attributes, serving as a basis for comparison against others' Key vectors. The Value vector encapsulates the input token's information, which will be utilized in the final output of the self-attention layer.

To compute the attention scores, we take the dot product of the Query vector with the Key vector of all other tokens and normalize it by dividing by the square root of the dimensionality of the Key vectors ($\sqrt{\text{head-size}}$). This normalization is crucial as it prevents the softmax function, which is applied subsequently, from having extremely small gradients during backpropagation due to large variance in the unnormalized scores.

The self-attention process can be efficiently computed in a parallelized manner using matrix operations. The matrix of query vectors $\mathbf{Q}$ is multiplied by the transpose of matrix of keys $\mathbf{K}$, followed by division by $\sqrt{\text{head-size}}$, to produce the matrix of attention scores. A softmax function is then applied to each row of this matrix to obtain a probability distribution that represents how much focus each element should put on every other element. The resulting matrix is multiplied by the matrix of values $\mathbf{V}$, yielding the output of the self-attention layer, which is then passed to subsequent layers in the network:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\text{head-size}}}\right)\mathbf{V}$$

This output carries a weighted representation of each token, considering the entire sequence (see diagram 1.3).

1.3.4 Multi-Head Attention Mechanism

In the original Transformer model, the concept of multi-head attention is a core feature that allows the network to capture various aspects of information at different positions. Traditional attention mechanisms use a single set of queries, keys, and values to generate one attention-weighted output per position. However, the Transformer enhances this by implementing multiple attention "heads"

Rather than performing a single attention function with d_{model}-dimensional keys, values, and queries, the model projects the queries, keys, and values h times with different learned linear projections into d_k, d_k, and d_v dimensions, respectively. These projections are performed using parameter matrices W_i^Q, W_i^K, and W_i^V for $i = 1, \ldots, h$. Here, h represents the number of heads, and d_k and d_v are the dimensions of the keys and values for each head (see Figure Figure 1.4).

"Before we work on artificial intelligence why don't we do something about natural stupidity? " Steve Polyak.

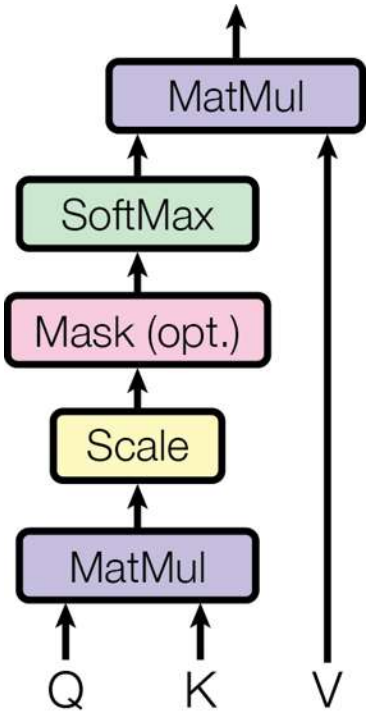

Figure 1.3: The self-attention mechanism in the Transformer architecture involves scaling the dot product of Query and Key vectors by the square root of the Key vector's dimensionality. An optional masking step can be applied. The scaled scores are then passed through a softmax function to convert them into a probability distribution. Then we calculate dot product of the result to the value vector. Taken from (Vaswani et al., 2017)

After projection, the attention function is applied in parallel to each set of d_k and d_v-dimensional keys, values, and queries, yielding d_v-dimensional output vectors. The outputs from all heads are then concatenated and linearly transformed with another learned projection, W^O, to produce the final values. This process can be formalized as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{1.3}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{1.4}$$

Multi-head attention enables the model to capture information from different representation subspaces. If a single attention head were used, the model would be restricted to averaging attention across all positions, which could potentially dilute the representation power. By using multiple heads, the network can maintain rich, position-specific representations by combining insights from different perspectives. This multi-faceted approach allows the Transformer to integrate various aspects of the sequence more effectively, leading to better performance on complex tasks. (Karpathy, 2023)

"A year spent in artificial intelligence is enough to make one believe in God. " Alan Perlis.

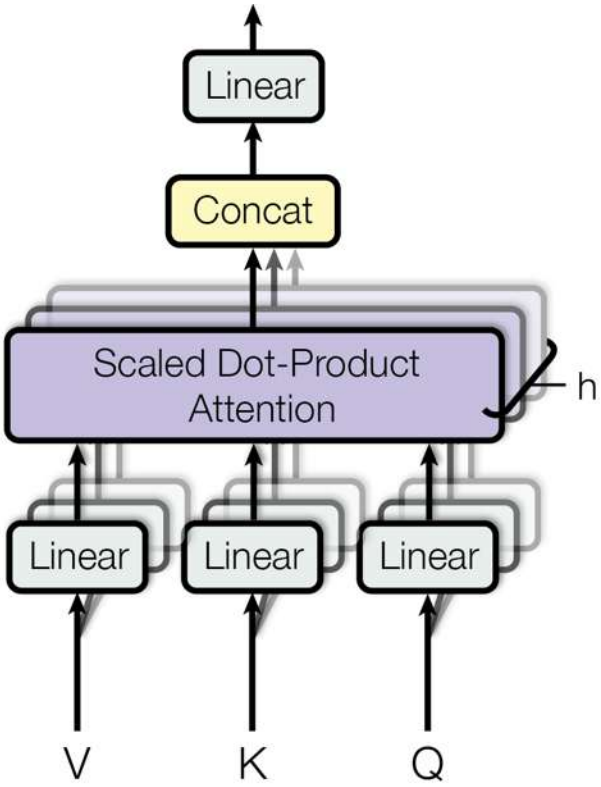

Figure 1.4: The multi-head attention mechanism in the Transformer model involves projecting queries, keys, and values multiple times with different learned linear projections. Each set of projections undergoes the attention function in parallel, producing output vectors. These outputs are concatenated and linearly transformed to yield the final values. Taken from (Vaswani et al., 2017)

1.4 LLMs

LLMs have significantly evolved over time, becoming more sophisticated with each iteration. Initially, language models were smaller and had limited abilities, but with advancements in computational power and machine learning techniques, these models have grown in scale. LLMs today are capable of understanding and generating human-like text, performing complex reasoning, and covering expert knowledge across various domains. Their training involves pretraining on vast corpora of text in a self-supervised manner, followed by fine-tuning to align with human preferences using techniques like Reiforcement Learning From Human Feedback (RLHF). This process, while computationally intensive, has led to LLMs being adopted for a wide range of applications, making them integral to advancements in AI. (Zhao et al., 2023)

"Artificial intelligence is the science of making machines do things that would require intelligence if done by humans. " John McCarthy.

1.5 Llama2 Overview

Llama2 represents a significant advancement in language modeling. Building on the foundations laid by earlier models, Llama2 incorporates novel architectural improvements and training techniques that enhance its performance and utility.

The architecture of Llama2 is based on an optimized transformer, with specific enhancements such as increased context length and grouped-query attention mechanisms that contribute to its efficiency and scalability.

1.5.1 Training Data

Llama2 was pre-trained on an extensive and robust corpus of self-supervised data. Efforts have been made to ensure the quality and factual accuracy of the training data, which ultimately contributes to the model's reduced propensity to generate inaccurate information or "hallucinations"

1.5.2 Performance Benchmarks

Performance benchmarks indicate that Llama2 models, particularly those fine-tuned for chat applications, perform favorably when compared to existing open-source models and even some closed-source competitors.

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
MPT	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
Falcon	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
LLAMA 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
LLAMA 1	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
LLAMA 1	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
LLAMA 1	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
LLAMA 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
LLAMA 2	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
LLAMA 2	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
LLAMA 2	70B	37.5	71.9	63.6	69.4	35.2	68.9	51.2	54.2

Table 1.1: Overall performance on grouped academic benchmarks compared to open-source base models.

1.5.3 Model Requirements

For our work, we decided to use Llama2-7b because it provides reasonably good performance and can be trained in a single GPU since it is the smallest model of the Llama series.

The computational requirements for operating Llama2-7b vary depending on the precision of the parameters and the phase of the machine learning pipeline (inference or training). For inference, the model operates under different precision levels: full precision (float32), half precision (float16), and more compressed forms such as 8-bit and 4-bit representations.

"The biggest risk is not taking any risk... In a world that is changing really quickly, the only strategy that is guaranteed to fail is not taking risks. " Mark Zuckerberg.

In full precision (float32), each parameter of the model requires 32 bits, or 4 bytes, to be stored. Thus, the storage requirement for the complete model is calculated as follows:

$$4 \text{ bytes/parameter} \times 7 \text{ billion parameters} = 28 \text{ billion bytes} = 28 \text{ GB of GPU memory.}$$

When utilizing half precision (float16), the storage is reduced by half, requiring 16 bits or 2 bytes per parameter. The total memory needed for inference in this precision is:

$$2 \text{ bytes/parameter} \times 7 \text{ billion parameters} = 14 \text{ GB of GPU memory.}$$

For even more compressed representations, such as 8-bit or 4-bit algorithms, the memory requirements decrease further. With 4-bit quantization, each parameter requires half a byte, which amounts to:

$$0.5 \text{ bytes/parameter} \times 7 \text{ billion parameters} = 3.5 \text{ GB of GPU memory.}$$

The memory requirements for training are contingent on the chosen optimizer. A commonly used optimizer is AdamW, which requires storage for the parameters, gradients, and second-order gradients. This typically equates to 8 bytes per parameter, resulting in:

$$8 \text{ bytes/parameter} \times 7 \text{ billion parameters} = 56 \text{ GB of GPU memory.}$$

Alternatively, the AdaFactor optimizer requires only 4 bytes per parameter, or 28 GB of GPU memory. The 8-bit optimizers from bitsandbytes, such as 8-bit AdamW, need only 2 bytes per parameter, corresponding to 14 GB of GPU memory.

In our thesis, we explored three distinct model sizes by fine-tuning Llama2-7b with 4-bit and 8-bit quantization as well as without any quantization. Although AdamW was used as the optimizer, the implementation of LoRA (Low-Rank Adaptation) means that not all gradients are loaded simultaneously. Instead, only the gradients for the trainable parameters, which comprise approximately 0.062% of the total as indicated in Table 1.2, need to be accounted for. Consequently, the memory requirements for training Llama-2-7b are considerably reduced due to efficient quantization and parameterization techniques. According to (Molino & Addair, 2023), storing the state of the optimizer requires approximately 4GB. The storage needed for the LoRa parameters is around 1GB, and for the gradients, an additional 1GB is necessary. This totals to a memory requirement of $4GB + 1GB + 1GB = 6GB$ for these components alone. When we include the model parameters at different quantizations, the total memory requirement varies: for 4-bit quantization, the combined memory requirement is $6GB + 3.5GB = 9.5GB$; for 8-bit quantization, it is $6GB + 7GB = 13GB$; with half precision (float16), the requirement is $6GB + 14GB = 20GB$; and for full precision (float32), the total memory needed is $6GB + 28GB = 34GB$. These figures reflect the reduced footprint of LLaMA-2-7b, making it feasible for training on hardware with lower memory capacities.

"Artificial intelligence is not a substitute for human intelligence; it is a tool to amplify human creativity and ingenuity. " Fei-Fei Li.

Model	Trainable Parameters	Total Parameters	Percentage
LLaMA-2-7b	4,194,304	6,742,609,920	0.062%
Zephyr-7b-alpha	3,407,872	7,245,139,968	0.047%

Table 1.2: Trainable Parameters of LLaMA-2-7b and Zephyr-7b-alpha Models

1.5.4 Applications and Impact

Llama2 has broad applications in numerous natural language processing tasks. Its accessibility and advanced capabilities contribute significantly to academic research and practical applications in language understanding. Building upon the foundations laid by previous work, the present thesis extends the application of Llama2, specifically the 7B model variant, by fine-tuning it for translating natural language into first-order logic. This effort is inspired by other papers that have harnessed the power of the earlier version of Llama for such tasks (Yang et al., 2023). The advancement of translating natural language into formal logic representations can significantly enhance machine understanding of complex logical structures and reasoning capabilities.

The model's open-source nature facilitates transparency and fosters further innovation in the field. By allowing researchers to build upon and refine its capabilities, Llama2 sets the stage for the development of more sophisticated language models and the exploration of new problem domains in natural language understanding.

1.6 Fine-tuning

Fine-tuning stands at the forefront of deep learning advancements, offering a pragmatic approach to adapting pre-trained models for specialized tasks. This technique, a subset of transfer learning, involves minor yet strategic modifications to a model's internal parameters, previously trained on a vast dataset for a general task. The essence of fine-tuning lies in its ability to harness the foundational knowledge acquired by the model, such as image recognition or natural language understanding, and tailor it to excel in a new, related domain. The core architecture of the model largely remains unchanged; however, these adjustments are pivotal in optimizing the model's performance for a specific task.

1.6.1 Why Embrace Fine-Tuning?

Efficiency and Resource Optimization

Developing a deep learning model from the ground up is a resource-intensive endeavor, marked by substantial time and computational demands. Fine-tuning, conversely, presents a cost-effective alternative that leverages the groundwork laid by pre-trained models. This approach significantly diminishes the necessity for extensive training periods, allowing researchers to bypass the rudimentary phase of learning and concentrate on refining the model for the task at hand.

"Robots are not going to replace humans, they are going to make their jobs much more humane. Difficult, demeaning, demanding, dangerous, dull – these are the jobs robots will be taking. " Sabine Hauert.

Enhanced Performance through Pre-Learned Features

Pre-trained models are reservoirs of valuable features and patterns, distilled from extensive datasets. Fine-tuning taps into this repository of knowledge, enabling the adapted model to perform with improved accuracy and efficiency on specialized tasks. This transfer of learning is instrumental in achieving superior outcomes, often surpassing what could be attained from training solely on a task-specific dataset.

Data Efficiency in Scarce Environments

The scarcity of labeled data for specific tasks poses a considerable challenge in the field of machine learning. Fine-tuning offers a viable solution by maximizing the utility of limited labeled data. By starting with a model that is already versed in general features, researchers can achieve commendable results with a fraction of the dataset otherwise required, making fine-tuning an invaluable tool in data-constrained scenarios.

1.6.2 The Fine-Tuning Process: A Step-by-Step Guide

Selection of a Pre-trained Model

Our journey in fine-tuning commenced with the selection of Llama2 (detailed in Section 1.4), a model whose foundational training and capabilities align closely with the objectives of our project.

Architectural Adjustments and Layer Freezing

Adapting Llama2 for our specific needs entailed architectural modifications, particularly in the model's upper layers, to better suit our task's unique requirements (explored in section 1.5.3). This process included decisions on freezing or unfreezing layers, a critical step in determining which parts of the model should remain static to retain general knowledge, and which should adapt to learn from our task-specific data.

Training with a Task-Specific Dataset

The training phase involved careful calibration with an appropriate dataset for that; for our thesis, the dataset used is detailed in section 4.1. There are some key steps when fine-tuning, including the implementation of a reduced learning rate to gently steer the pre-learned features toward our specialized task without undermining the model's existing knowledge base.

In conclusion, Fine-tuning represents a strategic inflection point in the pursuit of specialized model performance; In the context of our project, we employed the LoRa technique within the fine-tuning framework described in the next section. We applied this technique within a Ludwig (documentation of ludwig can be found in appendex A) configuration as described in section 4.2.

"I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted. " Alan Turing.

1.6.3 LoRa

Decomposition of ΔW

Fine-tuning a neural network often involves modifying its weight matrix, W, for a new task. These modifications, represented by ΔW, result in an updated weight matrix $W + \Delta W$. LoRA proposes a different approach by decomposing ΔW to reduce computational overhead in fine-tuning.

Intrinsic Rank Hypothesis

The intrinsic rank hypothesis asserts that changes to the neural network can be represented in a lower-dimensional space. It suggests that a subset of the changes in ΔW can capture the necessary adjustments effectively.

Matrices A and B

LoRA introduces two lower-rank matrices, A and B, such that $\Delta W = BA$. This leads to an updated weight matrix:

$$W' = W + BA \tag{1.5}$$

In this approach, W is not modified during training. The product of B and A, with lower dimensions, serves as a low-rank approximation of ΔW.

Impact on Trainable Parameters

By using matrices A and B of lower rank, the number of trainable parameters decreases. If W is a $d \times d$ matrix, traditionally updating W involves d^2 parameters. With B and A sized $d \times r$ and $r \times d$, the total parameter count is $2dr$, lower than d^2 for $r < d$.

This approach reduces the memory footprint, speeds up training and adaptation, enables fine-tuning on less powerful hardware, and facilitates scaling to larger models without a proportional increase in computational resources.

In the forthcoming chapters, we will delve into the intricacies of First-Order Logic and unpack the comprehensive details surrounding the training methodologies and overarching strategies employed in our thesis as well as the incorporation of Z3 to achieve the reasoning task.

Final Remarks

We have traversed the foundational and advanced aspects of neural networks and their application in natural language processing. We began by exploring the core principles of neural networks, including their architecture, activation functions, and optimization algorithms. We delved into the impact of the Transformer architecture, with its self-attention and multi-head attention mechanisms, which have revolutionized the field by enabling models to process sequences with greater context and parallelization capabilities. Finally we discussed LLMs, specifically Llama2 and fine-tuning techniques like LoRa. For the work of our thesis we are going to fine-tuned Llama2 for translating natural language to first order logic. In the next chapter we'll discuss fisrt order logic.

"AI is a mirror, reflecting not only our intellect, but our values and fears. " Ravi Narayanan.

Chapter 2

The Pillars of Logic: Understanding First-Order Logic

2.1 Introduction to First-Order Logic

First-Order Logic (FOL), also known as First-Order Predicate Logic, is a formal system used to represent and reason about propositions involving objects, relationships, and properties. It is a foundational component of mathematical logic and plays a pivotal role in various disciplines such as mathematics, computer science, and philosophy. This prominence isn't accidental; for those well-versed in modern logic, the emergence and adoption of FOL can seem natural and inevitable.

2.2 Syntax of FOL

In FOL, statements are constructed using a formal language comprised of variables, constants, predicates, functions, quantifiers, logical connectors, and parentheses. Each of these components plays a crucial role in formulating sentences in this formal system. In this chapter, we will delve into the specifics of each component to understand how they contribute to the construction of FOL sentences.

2.2.1 Symbols

Symbols in FOL can be broadly categorized into two groups: vocabulary symbols and logic symbols.

Logic Symbols

Quantifiers: Quantifiers specify whether a statement holds for some or all objects within a domain. While we will delve deeper into their meanings in the semantics section, it is worth noting here the two primary quantifiers:

- Universal $\forall$.

"Contrariwise, if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic." Lewis Carroll.

- Existential $\exists$.

Variables: Variables refer to non-specific objects within the domain. They are typically quantified by either $\exists$ or $\forall$ Variables that are not quantified are termed "free variables", leading to the concept of sentences which will be addressed later. There can be infinitely many variables (countably infinite, to be precise) they are commonly represented by letters such as x, y, and z, occasionally with subscripts for differentiation.

Connectives: Connectives enable the creation of compound formulas by linking two or more simpler formulas. Common connectives include:

- Implication: $\rightarrow$
- Conjunction: $\wedge$
- Disjunction: $\vee$
- Biconditional (Double implication): $\leftrightarrow$
- Exclusive disjunction: $\oplus$
- Negation: $\neg$

Contradiction, Tautology: Used respectively to denote logical contradictions and tautologies, these symbols are:

- $\bot$
- $\top$

Equality Symbol: Denoted by " = ", this symbol represents syntactic equality between expressions. While it's essential in many mathematical contexts, it may not be obligatory in other FOL scenarios.

Parentheses: Just as in arithmetic, parentheses in FOL determine the precedence of operations or connectors. They come in various forms: ( ), [ ], { }.

Vocabulary Symbols

Each first-order theory possesses its unique vocabulary. This vocabulary differentiates one theory from another, determining its specific domain or subject of discourse. This means that if for example, we are in the domain of set theory some symbols apply, different symbols may apply in the domain of number theory, or the domain of planets. The definition of the vocabulary is particular to the discourse but there are classification of the vocabulary symbols as explaind here:

Individual Constants: Constants represent specific objects. For example, in a theory discussing numbers, '0' or '1' might be treated as constants. In set theory, the symbol $\emptyset$ that represents the empty set is a constant too. In the context of natural language, a constant is just a proper name such as: *Alice*, *EiffelTower*, *Mexico*, *AmazonRiver*, *Jupyter*.

"A mind all logic is like a knife all blade. It makes the hand bleed that uses it." Rabindranath Tagore.

Predicate Symbols: These symbols are used to denote properties. It is not uncommon to represent predicates with singular letters such as Q,P, with a certain meaning. For instance, if P stands for "being an even number", then $P(x)$ means that x possesses the property of being even.

It is also common to represent predicates more descriptively, especially when it comes to applied or real-world scenarios, one might encounter predicates like $Planet(x)$, $ProfessionalAthlete(x)$, $LivingOrganism(x)$. In all cases, x is a variable we attribute to the property.

Relational Symbols: These symbols elucidate relationships between variables or constants. Mathematical contexts frequently employ relational symbols to express order, using notations like $<, >, \geqslant, \leqslant$.Symbols such as $\equiv, \sim, \approx, \simeq$ are used for equivalence relations. For set theory, symbols like $\in, \subset$ are prevalent. In real-world contexts, examples might include $Orbits(earth, sun)$ suggesting the Earth orbits the Sun, $LocatedIn(monalisa, louvremuseum)$ indicating the Mona Lisa is housed in the Louvre Museum. It is important to mention that some authors consider predicate symbols as relation symbols of arity 1, which means a relation symbol that admits only one parameter.

Functional Symbols: Representing operations or functions, functional symbols accept one or more arguments and yield a result. In the realm of mathematics, they often correspond to arithmetic processes like addition, subtraction, multiplication, or division. These symbols aid in defining functions that manipulate values, producing new outcomes based on designated rules. It is a bit more difficult to define a functional symbol in a natural context. Consider s as a function transforming a noun into its plural form; thus, $s(person)$ would be people.

2.2.2 Terms

Having familiarized ourselves with the essential symbols in First-Order Logic, the next logical step is to understand how these symbols can be seamlessly combined to form meaningful statements. Just as words in natural language come together to form sentences with specific semantics, in FOL, symbols are arranged into terms and formulas to represent complex ideas and relationships.

The **terms** in a language L can be thought of as the "nouns" of the language. They represent objects in our domain of discourse. They are defined by the following rules:

- If x is a variable, then x is a term.
- If c is a constant then c is a term.
- If f is a function symbol of arity $k > 0$ and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term.

"Illogical logic and unreasonable reasoning is how the truth becomes untruth." Andrew Cervantes.

2.2.3 Formulas

If terms are the nouns, then formulas are the statements we construct about those nouns. Predicates are used within formulas to attribute properties to terms or to describe relationships between them.

A **formula** in a language L is defined by the following rules:

- If R is a relational symbol of arity $k > 0$ and $t_1, \ldots, t_k$ are terms, then $R(t_1, \ldots, t_k)$ is a formula. Such formulas are called **atomic formulas** (also in mathematics context $=$ is a common atomic formula so-called equation).

  The following are examples of sentences in natural language that can be translated into an atomic formula:

  - "The Eiffel Tower is a famous landmark"
    $FamousLandmark(eiffelTower)$
  - "Usain Bolt won gold in the Olympics"
    $WinsGold(UsainBolt, olympics)$
  - "George likes music"
    $LikesMusic(George)$

- If ψ and φ are formulas and x is a variable, then the following are formulas: $\bot$ (absurdity), $\top$ (tautology), $\neg\varphi$ (negation), $\psi \to \varphi$ (implication), $\psi \wedge \varphi$ (conjunction), $\psi \vee \varphi$ (disjunction), $\forall x\psi$ (universal quantification), $\exists x\varphi$ (existential quantification).

  We can state formulas with the atomic ones, for instance:

  - "The Eiffel Tower is a famous landmark in Paris".
    $FamousLandmark(eiffelTower) \wedge In(eiffelTower, paris)$
  - "An athlete who wins gold in the Olympics is considered an Olympic champion".
    $\forall x(Athlete(x) \wedge WinsGold(x, olympics) \to OlympicChampion(x))$
  - "If George likes music, he wants to compose."
    $LikesMusic(George) \to WantsToCompose(George)$
  - "Events are either happy or sad".
    $\forall x(Event(x) \to Happy(x) \oplus Sad(x))$
  - "At least one event is happy".
    $\exists x(Event(x) \wedge Happy(x))$
  - "everybody likes someone".
    $\forall x \exists y(Like(x, y))$

Additionally, some formulas can be seen as defined constructions, for instance:

- $\neg\varphi := \varphi \to \bot$
- $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$
- $\varphi \oplus \psi := (\varphi \vee \psi) \wedge \neg(\psi \wedge \varphi)$.

"Logic is like the sword - those who appeal to it, shall perish by it." Samuel Butler.

Free and bound variables

In First-Order Logic, variables can occur in two distinct contexts: as free variables or as bound variables. This distinction is crucial for understanding the meaning of formulas and their components.

A variable is said to be free within a formula if it is not within the scope of a quantifier. These variables do not refer to a specific entity in the domain of discourse; instead, they can be seen as placeholders for any object. In a way, formulas with free variables can be thought of as having "incomplete" information. In a mathematical context, we might encounter the formula $x + 1 = y$ in this formula it is not possible to say if it is true or false, it appears to be something to solve (similar to asking: are there any x, y that satisfy the equation?). For another example, let us say that we are talking about a company and the language has the predicate symbols $Worker$ and $WorkWith$. In the formula $\forall x((Worker(x) \land Worker(y)) \rightarrow WorkWith(x, y)))$, we do not know who is $'y'$, it is a variable without any further information, in this sense, it is impossible to say whether the formula is true or false and again it aims to be a question rather than a sentence, is there someone who works with everyone?.

Conversely, a variable is bound when it is under the scope of a quantifier. These variables are "tied" to the quantifier and have their reference fixed by it. When a variable is bound, the formula makes a more definitive statement about the domain. In the same examples $\forall x \forall y(x + 1 = y)$ we can say it is false now in the context of real numbers. And in the formula $\forall x \exists y((Worker(x) \land Worker(y)) \rightarrow WorkWith(x, y))$ we can say it is true, considering that nobody works alone in the company.

Sentences

Building on the concepts of free and bound variables, we arrive at the notion of a sentence in First-Order Logic. A sentence is a formula with no free variables; all variables within it are bound by quantifiers. Sentences are of particular interest because they make complete statements about the domain of discourse that are unambiguously either true or false. They are the fundamental building blocks for expressing knowledge and making assertions in a logical system.

The importance of sentences cannot be overstated. They form the basis for logical deductions, allow us to articulate axioms within a theory, and can be used to make inferences about the world. In essence, sentences are the vehicles through which we can represent and reason about information precisely and unambiguously. (Badesa, Jané, & Jansana, 1998)

2.3 First-Order Logic Semantics

The semantics of FOL constitute a fundamental pillar governing how statements within the logic are interpreted, evaluated, and assigned truth values. At its core, FOL is designed to provide a systematic framework for reasoning about complex relationships and properties within a given domain of discourse. Understanding the semantics of FOL involves delving into the intricate process of assigning meaning to its constituent elements, including constants, predicates, functions, quantifiers, and variables.

"No rational argument will have a rational effect on a man who does not want to adopt a rational attitude." Zahid Abas.

The essence of FOL semantics lies in the establishment of an interpretation, which delineates the context within which logical statements are evaluated. This interpretation involves defining a domain of discourse, which encapsulates the universe of all conceivable objects relevant to the logical statements at hand. This domain can range from mathematical entities like numbers and sets to real-world entities such as people, objects, and concepts.

Within this domain, the semantics of FOL bestows meaning upon the logical elements. Constants, representing specific objects within the domain, are endowed with their respective denotations. Predicates, which encapsulate relationships and properties, are given interpretations that define how they map to subsets of the domain. Functions, akin to mathematical operations, are endowed with functionalities that describe how they transform inputs into outputs.

Crucially, quantifiers form a cornerstone of FOL semantics, enabling the specification of statements that hold universally or existentially over the domain. The universal quantifier ($\forall$) asserts that a statement holds true for all objects in the domain. In contrast, the existential quantifier ($\exists$) asserts that there exists at least one object for which the statement is true. These quantifiers provide a means to express generalizations and assertions about the entire domain or specific subsets.

The truth value of a statement in FOL hinges on its interpretation and the assignment of values to variables within the logical expressions. Variables are placeholders that can be instantiated with specific objects from the domain. The truth or falsity of a statement is evaluated based on the satisfaction of predicates and functions, and the compatibility of these predicates and functions with the assigned values to variables.

In summary, the semantics of FOL imbue logical statements with meaning, enabling us to reason about complex relationships and properties within a defined domain of discourse. By defining interpretations, assigning meanings, and evaluating truth values, FOL semantics lays the groundwork for rigorous and structured reasoning, forming an essential bedrock for the application of logic in various domains.

While the concepts of interpretation and semantics in first-order logic are formal and involve detailed mathematical definitions and structures, we can attempt to grasp their essence more intuitively. At its core, an interpretation is like a mapping or a translation guide, we shall examine some examples to better understand. (Tarski, 1999)

Examples of Atomic Formulas and their Interpretations:

1. **Mathematical Domain:**
    - **Domain of Discourse:** Natural numbers, $\mathbb{N}$
    - **Atomic Formula:** $P(x)$ where P is a predicate symbol.
    - **Interpretation:** Let $P(x)$ mean "x is even."
    - Under this interpretation, the atomic formula $P(2)$ is true, whereas $P(3)$ is false.

"The beauty of mathematics only shows itself to more patient followers." Maryam Mirzakhani.

It is worth mentioning that in most mathematical contexts there are just a few actual predicates, for our porposes we will consider natural sentences as predicates in all cases but the reader should keep in mind that properties usually are constructions of formulas. In the example above $P(x)$: "x is even" would not be defined as a predicate but as an abbreviation of the formula: $\exists y(2 \times y = x)$ where $\times$ is the functional symbol of the product and 2 is the constant for the number two. Actually in a strict way in the Peano arithmetic theory, there is only one constant 0: "zero", and only one functional symbol $s(x)$: "successor of x" each subsequential number is defined by the s of the previous one and the operations of sum and product are defined by recursion of the operator s. In this particular case the sentence $P(2)$ is true because there exists the number 1 such that $1 \times 2 = 2$, for this argument we are using the inference rule explore in section 2.3.2 known as *Existencial Generalization*. In general when it comes to mathematics, the concept of "truth" is often based on an axiomatic system. A statement or formula is considered true if it can be derived from a set of axioms using established logical rules.

2. **Real-world Domain (Zoology):**

   - **Domain of Discourse:** Animals
   - **Atomic Formula:** Mammal(y)
   - **Interpretation:** Let Mammal(y) indicate that "y is a mammal."
   - If our domain includes creatures like "Attina the whale", "Hera the eagle',' and "Tucker the dog" then the atomic formulas Mammal(Attina) and Mammal(Tucker) are true, while Mammal(Hera) is false.

   In the case of Real-world domains like this one, there are no axioms and the concept of truth relies on the reality of the objects (that could be based on the hypothetico-deductive system depending on the science). In contrast with math, there might be plenty of predicates defined in a Real-world domain.

3. **Relational Domain (Geography):**

   - **Domain of Discourse:** Cities and countries
   - **Atomic Formula:** LocatedIn(c, t)
   - **Interpretation:** Let LocatedIn(c, t) indicate that "city c is located in country t."
   - Assuming our domain recognizes "Paris" and "France" as valid entities, the atomic formula LocatedIn(Paris, France) is true.

In addition to the fundamental components mentioned earlier, the semantics of First-Order Logic extends to the evaluation of compound formulas, which are constructed through the use of logical connectors such as conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$), and negation ($\neg$). These connectors serve as the building blocks for creating complex logical expressions that capture relationships, conditions, and implications.

"The only way you gain mental toughness is to do things you're not happy doing. If you continue doing things that you're satisfied and make you happy, you're not getting stronger. You're staying where you're at. Either you're getting better, or you're getting worse. You're not staying the same." David Goggins.

To understand the semantics of these connectors, we can refer to their respective truth tables. These tables indicate how a composed formula is evaluated based on the truth values of its constituent formulas, we can consider each atomic formula as either true (T) or false (F):

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

Table 2.1: Truth table for disjunction ($\vee$)

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

Table 2.2: Truth table for conjunction ($\wedge$)

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Table 2.3: Truth table for implication ($\rightarrow$)

In essence, the semantics of First-Order Logic encompasses both the interpretation of individual elements within the logic and the evaluation of compound formulas composed through logical connectors.

2.3.1 Logical Equivalence

In the vast universe of logical formulas, it's possible to construct an infinite number of expressions involving the subformulas P and Q, along with various combinations of logical connectors. However, in the structure of the truth tables, we can see there are $2^{2^2} = 2^4 = 16$ possible outcomes for two formulas connected, a key observation is that no matter how complicated or intricate a formula might appear if it is constructed solely from P and Q, then its truth behavior must correspond to one of the 16 possibilities.

This brings us to the concept of logical equivalence. Two formulas are said to be logically equivalent if and only if they have the same truth value for every possible assignment of truth values to their atomic formulas.

The following list contains 16 different formulas in the sense of truth values (just like it is shown in Table 4), therefore every single formula built with P and Q is equivalent to some of those:

"The whole is more than the sum of its parts." Aristotle.

1. $O_1 : P$ (Identity)
2. $O_2 : Q$ (Identity)
3. $O_3 : \bot$ (Contradiction)
4. $O_4 : \top$ (Tautology)
5. $O_5 : P \wedge Q$ (And)
6. $O_6 : \neg P$ (Not)
7. $O_7 : \neg Q$ (Not)
8. $O_8 : P \vee Q$ (Or)
9. $O_9 : \neg(P \wedge Q)$ (Nand)
10. $O_{10} : \neg(P \vee Q)$ (Nor)
11. $O_{11} : P \oplus Q$ (Exclusive Or)
12. $O_{12} : P \leftrightarrow Q$ (Equivalence)
13. $O_{13} : P \rightarrow Q$ (Implication)
14. $O_{14} : Q \rightarrow P$ (Implication)
15. $O_{15} : \neg(P \rightarrow Q)$ (Inhibition)
16. $O_{16} : \neg(Q \rightarrow P)$ (Inhibition)

P	Q	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}
T	T	T	T	F	T	T	F	F	T	F	F	F	T	T	T	F	F
T	F	T	F	F	T	F	F	T	T	T	F	T	F	F	T	T	F
F	T	F	T	F	T	F	T	F	T	T	F	T	F	T	F	F	T
F	F	F	F	F	T	F	T	T	F	T	T	F	T	T	T	F	F

Table 2.4: All possible outcomes when combining two formulas

Classical Equivalences

Now let us explore some of the most commonly known equivalences in logic, we are using the symbol "$\equiv$" to represent that two formulas are equivalent.

- **De Morgan's Laws**:
  - Negation of a conjunction is equivalent to the disjunction of the negations:

$$\neg(P \wedge Q) \equiv (\neg P) \vee (\neg Q)$$

"Be yourself; everyone else is already taken." Oscar Wilde.

- Negation of a disjunction is equivalent to the conjunction of the negations:

$$\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q)$$

- **Tautology, and Contradiction**:
  - A proposition OR its negation is always true:

$$P \vee (\neg P)$$

  - A proposition AND its negation is always false (Contradiction):

$$P \wedge (\neg P)$$

- **Implication Equivalence**:
  - An implication can be represented using OR:

$$P \to Q \equiv (\neg P) \vee Q$$

- **Biconditional (Double Implication) Equivalence**:
  - The biconditional can be represented using conjunctions and implications:

$$P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$$

- **Distributive Laws**:
  - AND distributes over OR:

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

  - OR distributes over AND:

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

Quantifier Equivalences

Also, there are some very useful equivalences when it comes to quantifiers, but we have to be very careful with them since they could be tricky to identify. In general, the way to verify an equivalence is by using the inference rules that we will see in the next section.

- **Negation of Universal Quantifier:**

$$\neg\forall x\, P(x) \equiv \exists x\, \neg P(x)$$

  This states that it's not the case that every x has the property P if and only if there exists some x that doesn't have the property P.

"The greatest progress that the human race has made lies in learning how to make correct inferences." Friedrich Nietzsche.

- **Negation of Existential Quantifier:**

$$\neg\exists x\, P(x) \equiv \forall x\, \neg P(x)$$

This states that it's not the case that some x has the property P if and only if every x doesn't have the property P.

- **Switching Quantifiers:**

$$\forall x \forall y\, P(x, y) \equiv \forall y \forall x\, P(x, y)$$

This indicates that the order of quantifiers of the same species is interchangeable (the same is true for $\exists$ quantifier).

$$\forall x \exists y\, P(x, y) \not\equiv \exists y \forall x\, P(x, y)$$

This indicates that the order of different quantifiers matters and they are generally not interchangeable.

- **Distributivity over Quantifier:**

$$\forall x(P(x) \land Q(x)) \equiv (\forall x P(x) \land \forall x Q(x))$$

This states that for every x, both $P(x)$ and $Q(x)$ hold if and only if for every x, $P(x)$ holds and for every x, $Q(x)$ holds.

$$\exists x(P(x) \land Q(x)) \rightarrow (\exists x P(x) \land \exists x Q(x))$$

This states that if there exists an x such that both $P(x)$ and $Q(x)$ hold, then there exists an x for which $P(x)$ holds and another (possibly the same) x for which $Q(x)$ holds. Note that this is an implication, not an equivalence.

2.3.2 Inference Rules

FOL employs axioms and inference rules to facilitate the derivation of conclusions from given statements. These inference rules provide a formal mechanism for reasoning within the logic. Below are several fundamental inference rules in First-Order Logic:

- **Universal Instantiation (UI)**: If $\forall x : P(x)$ is true, then $P(c)$ is true for any constant c.
- **Existential Generalization (EG)**: If $P(c)$ is true for some constant c, then $\exists x : P(x)$ is true.
- **Universal Generalization (UG)**: If $P(c)$ is true for any constant c, then $\forall x : P(x)$ is true.
- **Existential Instantiation (EI)**: If $\exists x : P(x)$ is true, then $P(c)$ is true for a new constant symbol c.
- **Conjunction Introduction (CI)**: If P is true and Q is true, then $P \land Q$ is true.

"The simple believes every word, But the prudent considers well his steps." Proverbs 14:15.

- **Conjunction Elimination (CE)**: If $P \wedge Q$ is true, then P is true and Q is true.
- **Disjunction Introduction (DI)**: If P is true, then $P \vee Q$ is true.
- **Disjunction Elimination (DE)**: If $P \vee Q$ is true, and from P you derive R, and from Q you derive R, then R is true.
- **Implication Introduction (II)**: If from P you derive Q, then $P \to Q$ is true.
- **Implication Elimination (IE)**: If $P \to Q$ is true and P is true, then Q is true.
- **Negation Introduction (NI)**: If from P you derive a contradiction ($\perp$), then $\neg P$ is true.
- **Negation Elimination (NE)**: If $\neg\neg P$ is true, then P is true.(Miquel, 2017)

These inference rules are the foundation for drawing logical conclusions and making deductions within the First-Order Logic. By employing these rules alongside appropriate axioms, it becomes possible to systematically derive new statements and expand our understanding of the relationships and properties within the logical framework, also some formulas hold no matter the domain, for instance, all the equivalences we mentioned before.

To illustrate the application of inference rules, let's consider the logical statement:

$$\exists x(P(x) \wedge Q(x)) \to (\exists x P(x) \wedge \exists x Q(x))$$

We can prove this as follows:

1. **Assumption (for Conditional Proof)**:

   $$\exists x(P(x) \wedge Q(x))$$

2. **Existential Instantiation (E.I.)**: Given our assumption, there exists some constant c such that:

   $$P(c) \wedge Q(c)$$

3. **Conjunction Elimination (C.E.)**: From the above, we can deduce:

   $$P(c)$$

   And also:

   $$Q(c)$$

4. **Existential Generalization (E.G.)**: Given $P(c)$, we can infer:

   $$\exists x P(x)$$

   Similarly, from $Q(c)$, we infer:

   $$\exists x Q(x)$$

"A false conclusion once arrived at and widely accepted is not easily dislodged and the less it is understood the more tenaciously it is held." Georg Cantor.

5. **Conjunction Introduction (C.I.)**: Combining the two results from the previous step, we get:
$$\exists x P(x) \land \exists x Q(x)$$

6. **Implication Introduction (I.I.)**: Given our initial assumption and the final result we derived, we can state:
$$\exists x(P(x) \land Q(x)) \rightarrow (\exists x P(x) \land \exists x Q(x))$$

2.3.3 Satisfiability in FOL

A formula in First-Order Logic is considered *satisfiable* if there exists at least one interpretation (or model) in which the formula is true. This interpretation includes specific mappings for the variables, functions, and predicates within the formula to elements and relations in a particular domain.

- **Interpretation (Model):** An interpretation is a formal structure that assigns meaning to the symbols of a logical language. It consists of a domain of discourse (a set of objects) and interpretations of function symbols and predicate symbols in that domain.
- **Example:** Consider the FOL formula $\exists x P(x)$, which asserts that there exists an object x such that the predicate P holds for x. This formula is satisfiable if we can find at least one interpretation where $P(x)$ is true for some x in its domain. For instance, if our domain is the set of all humans and $P(x)$ is interpreted as "x is a philosopher," the formula is satisfiable if at least one philosopher exists.

Satisfiability tells us that a formula is logically possible, but it does not guarantee that the formula is true in every possible interpretation. A formula can be satisfiable even if it is not valid.

2.3.4 Validity in FOL

A formula in First-Order Logic is *valid* if it is true under all possible interpretations. In other words, no matter how we assign objects, functions, and predicates in the domain of discourse, the formula cannot be false. Valid formulas represent logical truths.

- **Tautology:** In propositional logic, a valid formula is often called a tautology. In FOL, the concept is similar but involves quantifiers and predicates/functions over a domain of discourse.
- **Example:** The formula $\forall x P(x) \rightarrow \exists x P(x)$ is valid. It states that if every object x satisfies predicate P, then there exists at least one object x that satisfies P. No matter how we interpret P or choose our domain, this statement cannot be false; if P holds for all objects, it must hold for at least one.

"Pure mathematics is, in its way, the poetry of logical ideas." Albert Einstein.

Validity is a stronger condition than satisfiability because it requires the formula to be true in all interpretations, not just one. It's worth noting that in our proof, we did not make use of any particular formula or assume a specific domain for our predicates P and Q. This means that our proof holds universally, across all theories and domains. The statement we proved is valid irrespective of the specific nature of P and Q, making it a tautology in the realm of first-order logic.

2.4 First-Order Logic Translation

2.4.1 First-Order Logic into Natural Language

To convert an FOL formula into Natural language is the process of reading correctly the symbols and predicates of a logical formula.

Here is a guide on how to read common logical symbols:

- $\forall$: Read as "for all" or "for every" followed by the variable it is quantifying. E.g., $\forall x(Person(x))$ is read as "for all x, x is a person".
- $\exists$: Read as "exists", followed by the varibale it is quantifying, followed by "such that". E.g., $\exists y(Planet(y))$ is read as "exist y such that y is a planet".
- $\wedge$: Read as "and". E.g., $Movie(m) \wedge IntriguingPlot(m)$ is read as "m is a movie and it has intriguing plot".
- $\rightarrow$: Read as "implies" or "then". E.g., $Tree(t) \rightarrow ProducesOxygen(t)$ is read as "if t is a tree then it produces oxygen".
- $\neg$: Read as "not". E.g., $\neg Dance(Jhin)$ is read as "Jhin does not dance" or "it is not the case that jhin dances".
- $\oplus$: Read as "either or" or "xor". E.g., $\forall x(Switch(x) \rightarrow (On(x) \oplus Off(x)))$ is read as "for all x, if x is a switch then x is either on or off".
- $\vee$: Read as "or". E.g., $\exists w(PlaySoccer(w) \vee \forall w(\neg PlaySoccer(w))$ is read as "exists w such that w plays soccer or for all w, w does not play soccer".
- $\leftrightarrow$: Read as "if and only if" or "is equivalent to". E.g., $Acid(a) \leftrightarrow pHLevelLessThan7(a)$ is read as "a is and acid if and only if a has pH level less than seven".
- Parentheses (): Used to group symbols and dictate the order of operations. The content inside the parentheses is typically read as a single unit.

Also is common to see existential and universal quantifiers together at the beginning of a sentence for instance:

The formula $\forall x \exists y((Mexican(x) \wedge Chile(y)) \rightarrow like(x, y))$ could be read as For all x, there exists a y such that if x is a Mexican and y is a chile, then x likes y.

The formula $\exists x \forall y((Mexican(x) \wedge Chile(y)) \rightarrow like(x, y))$ could be read as exists x such that for all y if x is a Mexican and y is a chile, then x likes y.

"I don't believe in empirical science. I only believe in a priori truth. The meaning of world is the separation of wish and fact. Either mathematics is too big for the human mind or the human mind is more than a machine." Kurt Gödel.

2.4.2 Natural Language into First-Order Logic

Translating natural language statements into first-order logic (FOL) is a structured process that requires careful attention to linguistic details. Below are the general guidelines for performing this translation:

1. **Identify the Quantifiers:** Words such as “all”, “every”, “some”, “exists”, “no”, and “any”, often indicate the presence of quantifiers. Universal quantifiers ($\forall$) are used for general statements, while existential quantifiers ($\exists$) indicate the existence of at least one instance.

2. **Determine the Domain of Discourse:** Define the set of all objects under consideration in the context of the statement. Variables will represent members of this set.

3. **Assign Variables:** Allocate variables to represent the entities mentioned in the natural language statement.

4. **Identify Predicates:** Convert properties or relationships expressed in the statement into predicates.

5. **Translate Logical Connectives:** Map conjunctions disjunctions like “and”, “or”, and negations like “not” to their logical equivalents ($\wedge$, $\vee$, $\neg$), and conditional phrases like "if... then" to implications ($\rightarrow$).

6. **Use Parentheses to Structure Expressions:** Utilize parentheses to indicate the scope and precedence within the logical expression.

For example, consider the following natural language statement and its translation into FOL:

Natural Language Statement: “Every student loves some book”.

FOL Translation:

$$\forall x(Student(x) \rightarrow \exists y(Book(y) \wedge Love(x, y)))$$

This translation asserts that for every entity x, if x is a student, then there exists some entity y such that y is a book and x loves y.

When translating, it is important to note that natural language often carries ambiguities that may not be present in FOL. Therefore, clarity and precision are imperative in the translation process.

2.4.3 Challenges in Translating FOL into Natural Language

Translating formal logic statements into natural language is not always straightforward due to the variety of ways in which the same logical idea can be expressed and the inherent ambiguity in natural language. Each logical symbol can correspond to a range of linguistic expressions, and some natural language constructs can be particularly confusing.

The following is a list of common misunderstandings and difficult words to interpret in a sentence.

“Translation is at best an echo.” George Borrow.

- **Conjunction ($\wedge$)**: While "and" is the common translation for $\wedge$, other phrases like "but" can also imply conjunction. For instance, "John is tall but not good-looking" translates to $Tall(John) \wedge \neg GoodLooking(John)$.
- **Implication ($\rightarrow$)**: The implication is often seen as "if...then", but it can manifest in subtler forms such as "Every house is a building" which suggests that if something is a house, then it is also a building ($\forall x(House(x) \rightarrow Building(x))$). Conditional phrases like "unless" or "as long as" also indicate an implication but require careful translation to capture the correct meaning.
- **Disjunction ($\vee$)**: The word "or" is commonly used to express disjunction, but in natural language, it can sometimes include the meaning of exclusivity, which is not present in the logical $\vee$. For example, "You can have tea or coffee" often implies a choice between tea and coffee, not the possibility of having both.
- **Exclusive Or ($\oplus$)**: When exclusivity is explicitly stated, "either... or..." is typically used to express the exclusive or ($\oplus$). For example, "Either you can have tea or you can have coffee, but not both" translates to $Tea \oplus Coffee$.
- **Biconditional ($\leftrightarrow$)**: Phrases such as "if and only if" or "just in case" correspond to the biconditional. But we can face definitions that represent a biconditional, for instance, "A device is a smartphone if it has a touchscreen interface, internet access, and can run applications for various purposes". it appears to be a conditional but it stands both implications if something meets the requirements it is a smartphone and also if something is a smartphone it has the properties $\forall x(Smartphone(x) \leftrightarrow (Device(x) \wedge TouchscreenInterface(x) \wedge InternetAccess(x) \wedge RunsApplications(x)))$
- **Negation ($\neg$)**: Negation can be expressed through various terms in natural language, including "not", "no", "never", and "nobody". For example, "No one is perfect" translates to $\neg\exists x(Perfect(x))$.

Expressive Power

While first-order logic serves as a powerful tool for formalizing mathematical concepts and finds widespread applications in computer science and various other fields when it comes to natural language sometimes it falls short in representing it. The following are examples of sentences that cannot be completely represented by FOL:

- **Predicate adverbial:** Jorge is running quickly. In this example is not convenient to analyze it as $(Running(Jorge) \wedge Quickly(Jorge))$, we might be tempted to express it as $(RunningQuickly(Jorge)$, and even though it is a correct expression this representation does not explicitly differentiate between the act of running and the adverbial modifier "quickly".
- **Relative adjective:** Jumbo is a small elephant. This example might be analyzed as $(Elephant(Dumbo) \wedge Small(Dumbo))$ but in this case 'small' is bound to the expected size of elephants therefore it is not an absolute quality but rather a quality dependent on a comparison to the typical size of elephants.

"Hate is only a form of love that hasn't found a way to express itself logically." Lil Wayne.

- **Quantification over properties:** Carlos has some characteristics of a good student ($\exists P(GoodStudentCharacteristics(P) \wedge P(Carlos))$. In higher-order logic, we can deal with properties of properties, which is required for this sentence since P would be both a term and a property. Being a good student should be a predicate in FOL but since we are talking about only 'some attributes' it requires quantifiers over predicates, which cannot be implemented in single-sorted first-order logic.

Final Remarks

We have explored the essential framework and intricacies of First-Order Logic (FOL), a cornerstone in the realm of formal logic. FOL's capacity to represent and reason about propositions involving objects, relationships, and properties.

We began by dissecting the syntax of FOL, delving into the symbols and structures that form the backbone of logical expressions. This understanding paved the way to explore the semantics of FOL, which imbues these symbols with meaning and allows us to evaluate their truth within a given domain of discourse. Through examples and truth tables, we clarified how logical connectors and quantifiers operate, ensuring precise reasoning in complex scenarios.

The chapter also highlighted key concepts such as logical equivalence and inference rules, which are vital for deriving conclusions and expanding our logical frameworks. We demonstrated how these rules underpin the process of logical deduction, allowing for systematic and rigorous reasoning.

Translating between natural language and FOL proved to be both an art and a science, requiring careful attention to linguistic nuances and logical precision. The challenges posed by ambiguities and the expressive limitations of FOL were acknowledged, highlighting areas where FOL falls short in capturing the richness of natural language. In the following chapter we will explore some tries in the literature for translating NL into FOL with automated systems using algorithms or leveraging the power of artifitial intelligence models.

"We are afraid of ideas, of experimenting, of change. We shrink from thinking about a problem through to a logical conclusion." Anne Sullivan.

Chapter 3

The Logic of Language: Exploring Systems in NL-to-FOL Translation and Reasoning

The field of Natural Language Processing has witnessed remarkable advancements over the years, leading to significant progress in tasks such as machine translation, sentiment analysis, and text generation. One crucial endeavor within NLP is the translation of natural language into formal and structured representations, such as First-Order Logic. This chapter embarks on an exploration of the current state of research and developments in the domain of translating Natural Language into First-Order Logic.

We will explore mainly three different works in this chapter:

- Rule-based translation by (Bansal, 2015)
- Large Language Model translation by (Yang et al., 2023)
- Classification among true, false and unknown premises' set and their conclusion by (Han et al., 2022b)

3.1 Part of Speech (POS) Tagging and Rule-Based Natural Language to First-Order Logic Translation

Within the realm of translating natural language into First-Order Logic, traditional algorithms have offered a structured approach by relying on sets of predefined rules and carefully designed algorithms. In this subsection, we explore the methodology by Namman Bansal (Bansal, 2015) which seeks to achieve NL to FOL translation through systematic rule-based processes. These approaches stand as predecessors to the more recent advancements in neural networks and machine learning and provide insights into how manual rule-based systems have historically been employed to bridge the gap between the expressive nature of human language and the formal constructs of logic.

"A philosopher once said, 'Half of good philosophy is good grammar'. " A. P. Martinich.

3.1.1 Understanding Part of Speech Tagging

Part of Speech tagging is a process in NLP where words in a sentence are marked with their corresponding part of speech. This categorization is based on both the definition of the word and its context within the sentence. POS tagging essentially disambiguates the syntactic and semantic role of each word, making it fundamental for understanding sentence structure. POS tagging helps in breaking down a sentence into its constituent parts, such as nouns, verbs, adjectives, etc (see Figure Figure 3.1). By analyzing the sequence and combination of these tags, the structure of a sentence can be determined. For instance, a simple sentence like "Men love apples" would be broken down into nouns (NN) for "Men" and "apples", and a verb (VB) for "love".

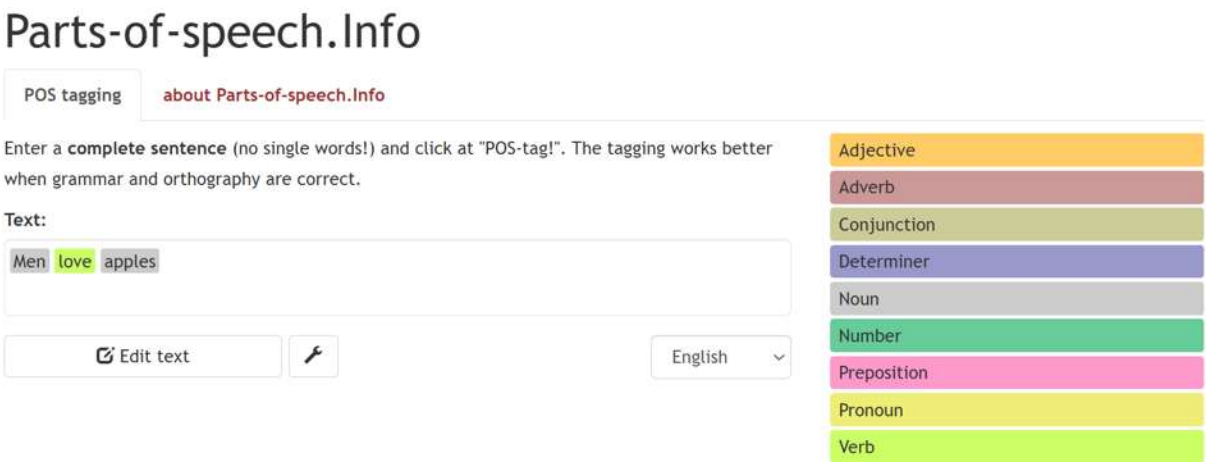

Figure 3.1: Example of POS tagging of the sentence "Men love apples", each color represents a differen part of the speech you can check the public page https://parts-of-speech.info/ for POS tagging:

The following table lists some but not all the POS tags used in the tagging process:

Tag	Description
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present

Table 3.1: Common POS tags and their descriptions

"The superior man is modest in his speech, but exceeds in his actions." Confucius.

3.1.2 Rules Based on Tag Structures

The translation of natural language to first-order logic in that thesis (Bansal, 2015) relies heavily on rules that are based on the structures identified through POS tags. These rules map the tag sequences to corresponding FOL structures. Due to the variability and complexity of language, a substantial set of rules is required to cover as many combinations and structures as possible. This ensures a more accurate and comprehensive translation process. The system employs two types of rules: non-recursive and recursive. Non-recursive rules are straightforward mappings, like "rule = (NN) VB (NN)" for basic sentences. Recursive rules, such as "rule = if S1 then S2", are used for more complex sentences that involve compounded logical structures. These rules are crucial for translating sentences with multiple clauses or conditional statements.

The following two tables show correct and incorrect translations by the system discussed in that thesis.

Natural Language	**First Order Logic Translation**
All humans eat some food	$\forall x(\text{human}(x) \rightarrow \exists y(\text{eat}(y) \wedge \text{food}(x, y)))$
Pen is mightier than sword	$\text{mightier}(\text{Pen}, \text{sword})$
Every gardener likes the sun	$\forall x(\text{gardener}(x) \rightarrow \text{likes}(x, \text{sun}))$
John looks for a unicorn	$\exists x(\text{unicorn}(x) \wedge \text{looks}(\text{John}, x))$
There is a barber who shaves every man	$\exists x(\text{barber}(x) \wedge \forall y(\text{man}(y) \rightarrow \text{shaves}(x, y)))$
Not every person who plays football knows cricket	$\neg\forall x((\text{person}(x) \wedge \text{plays}(x, \text{football})) \rightarrow \text{knows}(x, \text{cricket}))$
Some students are both intelligent and hard workers	$\exists x(\text{student}(x) \wedge (\text{intelligent}(x) \wedge \text{hardWorker}(x)))$

Table 3.2: Correct examples of translation in the Naman system

Natural Language	**First Order Logic Translation**
Julie feeds all her dogs	$\forall x(\text{feeds}(x, \text{Julie}) \rightarrow \text{dog}(\text{Julie}, x))$
All men are smart mortals	$\forall x(\text{manu}(x) \rightarrow \text{mortalu}(x))$
Pluto loves its master	$\exists x(\text{master}(\text{Pluto}, x) \wedge \text{loves}(x, \text{Pluto}))$
All flowers are not fragrant	$\neg\forall x(\text{flower}(x) \rightarrow \text{fragrant}(x))$
If something is damaged then the curator will be blamed	$\exists x(\text{damaged}(x)) \rightarrow \text{blamed}(\text{curator})$
Every person loves his mother	$\forall x\,(\text{person}(x) \rightarrow \exists y\,(\text{mother}(y, x) \wedge \text{loves}(x, y)))$
Either taxes are increased or if expenditure rises then the debt rises	$(\text{increased}(\text{taxes})) \vee ((\text{rises}(\text{expenditure})) \rightarrow \text{rises}(\text{debt}))$

Table 3.3: Inorrect examples of translation in the Naman system

"If you talk to a man in a language he understands, that goes to his head. If you talk to him in his language, that goes to his heart." Nelson Mandela.

3.1.3 Issues in the POS Tagging-Based Approach

While the POS tagging-based approach for NL-to-FOL translation offers a structured and rule-based method, it faces several potential issues:

1. **Complexity in Language Structure**: Natural languages often have complex and ambiguous structures that can be challenging to encapsulate fully with predefined rules. The variability and nuance in sentence construction could lead to incorrect or incomplete translations, for instance, the example shown in the 3.2 figure is quite complex and almost impossible for the ruler to detect "the main connector", it will look to see whether the sentence is an implication, a conjunction, or a disjunction; but given the complexity of the sentence it would probably fail.

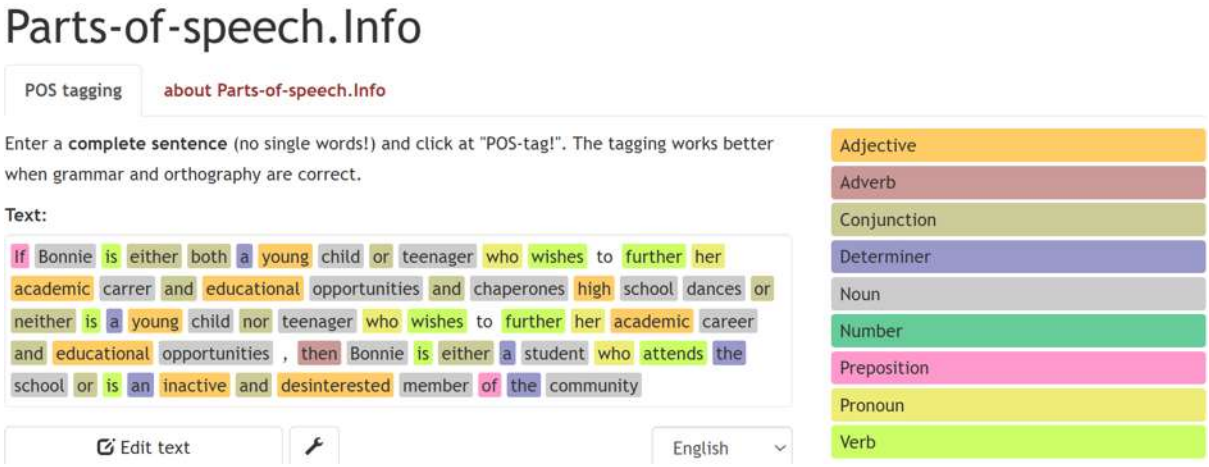

Figure 3.2: Complex natural language statement and its POS. The example was taken from the FOLIO data base we will discuss later in this chapter

2. **Limitation in Rule Flexibility**: The dependence on predefined rules may limit the system's ability to accurately translate sentences that do not fit neatly into these rules. Particularly, idiomatic expressions or sentences with implicit meanings might pose significant challenges. In the examples shown in Table 3.3 "Julie feeds all her dogs" was not correctly translated since it does not have the usual structure that the sentences must have according to the design of the algorithm.

3. **Scalability Concerns**: Expanding the rule set to cover a wider range of sentence structures could lead to scalability issues. As the number of rules increases, the system may become more complex and less efficient, particularly when dealing with long or compound sentences, in the words of Bansal, Naman:

> Optimal number of rules for such a system would be the number of distinct structure for First Order Logic we could have. We did clustering based on the First Order Logic structure on the benchmarks and found that we have 248 different structures for FOL in our benchmarks. We worked on an approach which targets partial sentence. Main focus was to reduce the number of rules in our system. We built a prototype based on initial ideas and we got some encouraging results. We had around 160 sentences being translated with around 25 rules.
>
> — (Bansal, 2015, p. 52).

"Divide each difficulty into as many parts as is feasible and necessary to resolve it." Rene Descartes.

4. **Handling Ambiguity**: Ambiguities inherent in natural language, such as polysemy or homonymy, can complicate the translation process. Determining the correct FOL representation for words with multiple meanings based on context can be challenging.

5. **Dependency on POS Accuracy**: The entire translation process heavily relies on the accuracy of POS tagging. Any errors in POS tagging could lead to incorrect translations, impacting the overall reliability of the system.

6. **Logical Connectors and Quantifiers**: Some specific structures of the sentences can confuse the algorithm given that the rules guide to identify some keywords with specific connectors or quantifiers, for instance in the example "All flowers are not fragrant" the combination of "not" and "all" lead the system to state $\neg\forall$ but that is incorrect in this particular case; in the example "If something is damaged then the curator will be blamed", since the word something is associated with $\exists$ the model gets confused in this particular case where it must be $\forall$. Furthermore this system does not handle correctly the exclusive or as in the example "Either taxes are increased or if expenditure rises then the debt rises".

3.1.4 Conclusions of the System

Their work (Bansal, 2015) concludes that a rule-based approach, utilizing POS tagging and carefully crafted bi-directional rules, can effectively translate NL sentences to FOL and vice versa. The approach aims to provide a learning tool for students, helping them understand the nuances of logical translation. The adaptability of the rule system is a key feature, allowing for continuous improvement and expansion to cover more complex or diverse sentence structures. Rule-based methods, offer a systematic and interpretable framework, making them suitable for educational purposes or applications where transparency is paramount. However, they often struggle with scalability and flexibility when faced with the diversity of natural language expressions. This is one of the reasons why in our translator we are not using this types of systems but instead leveraging the power of Large language models (LLMs).

3.2 LLM-Based FOL Translators

The endeavor to equip models with the capability to reason in text forms a cornerstone in the field of NLP, tracing back to its early days. Over the years, significant strides have been made in this domain. The emergence of various benchmarks has stimulated advancements in different facets of reasoning over textual inputs. These benchmarks encompass a wide spectrum of reasoning tasks, ranging from natural language inference (NLI) and commonsense knowledge reasoning to numerical, multi-hop, monotonicity, and knowledge-based reasoning. Within this diverse landscape, the focus on logical reasoning has gained increasing traction in recent years.

Several datasets have been developed to test various forms of logical reasoning, often in the context of standardized examinations, such as ReClor (Yu, Jiang, Dong, & Feng, 2020) and LogiQA (Liu et al., 2020). However, these datasets often mix different reasoning types and do not isolate logical reasoning. Meanwhile, the challenge of assessing logical reasoning independently has captured researchers' interest.

"Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better." Samuel Beckett.

Benchmarks like CLUTRR (Sinha et al., 2019), LTL (Tang & Belle, 2024), and others have explored inductive and temporal logic, and works such as RuleTaker (Clark et al., 2020) and LogicNLI (Tian et al., 2021) have delved into deductive reasoning with synthetic corpora.

Despite this extensive research landscape, our research narrows its focus to two pivotal contributions in the realm of translating natural language to first-order logic using Large Language Models: "FOLIO: A Dataset for NL-to-FOL Reasoning" (Han et al., 2022b) and "Harnessing Large Language Models for NL-to-FOL Translation" (Yang et al., 2023). These works stand out in the crowded field by their specific approach to leveraging LLMs for FOL translation, a task that encompasses both the complexity of natural language and the rigidity of formal logic. FOLIO, in particular, marks a significant milestone as the first in the most refined as of yet large-scale FOL reasoning dataset with formal logic annotations, characterized by its logical diversity and complexity. Besides it is the largest dataset written by humans in the field, it represents a nuanced and real-world informed approach to the challenge, setting it apart from the more synthetic or limited-scope datasets in the field.

3.3 "FOLIO: A Dataset for NL-to-FOL Reasoning"

3.3.1 Overall Approach

The FOLIO paper presents a dataset annotated with first-order logic to support the development of models for natural language reasoning. The dataset is structured to enable the translation of reasoning expressed in natural language into formal logic and back. To evaluate the effectiveness of various language models on this task, the authors of FOLIO undertook a series of experiments. These experiments involved the supervised fine-tuning of medium-sized language models, such as BERT and RoBERTa, to improve their reasoning capabilities within the framework of first-order logic (Han et al., 2022b).

Additionally, the paper explores the use of few-shot prompting techniques with larger language models, including GPT-NeoX (Black et al., 2022), OPT (Zhang et al., 2022), GPT-3 (Brown et al., 2020), and Codex (Chen et al., 2021). This experimentation aims to assess the potential of pre-trained language models to adapt to the specialized task of first-order logic reasoning when provided with a limited number of examples. Through systematic evaluation, the experiments seek to understand how well these models can apply their pre-existing linguistic knowledge to the specific challenges posed by the FOLIO dataset.

3.3.2 FOLIO dataset

The FOLIO dataset was meticulously developed, emphasizing quality and scalability through diverse methodologies and rigorous review processes. Expert annotators, including computer science undergraduates, graduate students, and senior researchers, contributed to its creation and review. The development process encompassed four key stages: example collection, NL-FOL alignment review, FOL annotation verification, and natural language review.

"Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom." Clifford Stoll.

Two primary methods were employed for example collection. **WikiLogic** involved creating stories from scratch based on real-world knowledge using random Wikipedia articles. This method resulted in a variety of topics, abundant natural language variations, and a broad vocabulary. In contrast, **HybLogic** (Hybrid Annotation) addressed the time-consuming nature of creating logically-sound stories from scratch. It used syllogisms to create logically-valid story templates, which annotators then filled with real-life scenarios, the overall estructure of the data set is seen in Figure 3.3.

During the **NL-FOL Alignment Review**, the alignment between NL sentences and FOL formulas was verified. This stage included adding necessary commonsense knowledge to both NL and FOL premises to ensure comprehensive premises for FOL reasoning.

FOL Verification involved checking the syntactic validity and label consistency of FOL formula annotations. FOL statements were converted into Python code snippets and checked using an adapted FOL prover.

The **Dataset Statistics** reveal that FOLIO comprises a significant number of stories, premises, and conclusions, featuring a diverse range of topics and a large vocabulary. It stands out for its logical complexity and diversity, with a much larger number of distinct Abstract Syntax Trees (ASTs) than previous datasets. An abstract syntax tree is a representation used in computer science to depict the structure of code or expressions. In this context, ASTs represent the logical structure of stories, premises, and conclusions within the dataset. This illustrates the dataset's capacity to support complex reasoning tasks.

conclusion string · *lengths* 12 386	premises sequence	premises-FOL sequence	label string 3 valu
Bonnie performs in school talent…	["If people perform in school…	["∀x (TalentShows(x) → Engaged(x))", "∀x (TalentShows(x) ∨ Inactive(x))", "∀x (Chaperone(x…	Unknow
If Bonnie is either both a…	["If people perform in school…	["∀x (TalentShows(x) → Engaged(x))", "∀x (TalentShows(x) ∨ Inactive(x))", "∀x (Chaperone(x…	True
If Bonnie either chaperones high…	["If people perform in school…	["∀x (TalentShows(x) → Engaged(x))", "∀x (TalentShows(x) ∨ Inactive(x))", "∀x (Chaperone(x…	False

Figure 3.3: FOLIO data set general structure, the image was taken from hugging face. The example in Figure 3.2 was taken from the conclusion column of this data set

3.3.3 Evaluation and Results

The evaluation of logical reasoning capabilities in the FOLIO paper revealed notable findings, Table 3.4 shows the accuracy of some models, the accuracy measures the number of successes in the label that can be: true when from the premises the conclusion is infered, false if the negation of the conclusion is infered,and unkwown if neither of them ocurred.

In addition to these results for accuracy using classification models, also some evaluations were performed using GPT-3 davinci and Codex davinci, focusing on several metrics for translating NL into FOL and then also getting the label: true, false or unknown. The

"Passion provides a purpose, but data drives decisions." Andy Dunn.

metric used for these models were: Syntactic Validity (SynV) that counts the number of correct FOL formulas generated by the model. Syntactic Exact Match (SynEM) that counts the percentage of sentences that were exactly the same as the FOL ground truth. Abstract Syntax Tree Match (AST) that percentage of sentences that are equal after calculating the AST of both the predicted and the ground truth. Predicate Fuzzy Match (Pr) the average score calculated using Levenshtein distance among predicted predicates and expected ones. Execution Accuracy (EAcc) that are the percentage of correct labels assigned by the model. The results are shown in Table 3.5.

Model	**Model Size**	**Acc (%)**
random probability	-	33.33
Fully supervised fine-tuning		
BERT-base	110M	56.83
BERT-large	340M	59.03
RoBERTa-base	110M	56.83
RoBERTa-large	340M	62.11
8-shot NL Prompt		
GPT-NeoX-20B	20B	38.77
OPT-13B	13B	30.75
OPT-66B	66B	40.53
GPT-3 text-curie-001	6.7B	34.36
GPT-3 text-davinci-001	175B	43.44
GPT-3 text-davinci-002	-	52.24
Codex code-davinci-001	-	37.80
Codex code-davinci-002	-	56.04

Table 3.4: Evaluation results for language models on FOLIO. The base case happens with a random model that predicts uniformly the labels and would have 33% of accuracy since the dataset is balanced. BERT-base and RoBERTa-base models demonstrated similar performances with an accuracy of 56.83%. An improvement was observed with BERT-large, which showed a 2.2% increase over BERT-base. Among these models, RoBERTa-large achieved the highest performance in the fine-tuning setting, with an accuracy of 62.11%.

Model	**SynV**	**SynEM**	**AST**	**Pr**	**EAcc**
GPT-3 davinci	94.55	47.35	62.18	58.92	46.87
Codex davinci	96.39	53.09	67.57	62.42	54.89

Table 3.5: Evaluation results for GPT-3 davinci and Codex davinci. They both achieved a good SynV showing the LLM capabilities of structuring correctly FOL formulas. There is a correlation between having a high performance of AST Pr and EAcc, which means having accurate formulas leads to better classification and having similar predicates also leads to better accuaracy

"True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information." Winston Churchill.

These results highlight the challenges faced by state-of-the-art (SoTA) Language Models (LMs) in reasoning with FOLIO, especially with longer and more complex reasoning chains. While GPT-3 and Codex demonstrated higher performance on examples with 1 to 4 premises, examples requiring 5 to 8 premises proved more challenging. This trend was also observed under the fine-tuning setting, where RoBERTa showed marginally better performance on examples with fewer premises. The FOLIO dataset, with its unique complexity and the prevalence of difficult examples, highlights the current limitations of LMs in extrapolating to longer and more intricate reasoning chains, suggesting a need for further study and improvement in this area.

3.3.4 Conclusions of FOLIO

The FOLIO study concludes that while the dataset offers a rich and complex testing ground for models' logical reasoning abilities, there is significant room for improvement, especially in the context of LLMs like GPT-3. The models generally performed poorly in predicting accurate truth values for False and Unknown conclusions, indicating a gap in their ability to handle complex logical reasoning tasks effectively (Han et al., 2022b).

The results obtained by our work fine-tuning Llama2 are comparable to FOLIO best performer which is RoBERTa-large and actually outperformed even this model in accuaracy. Our model also have the advantage of being able to translate and not only to classify true values.

To train Llama2 we are using FOLIO dataset along with other datasets as explained in 4.1

3.4 "Harnessing Large Language Models for NL-to-FOL Translation"

3.4.1 Overall Approach

In their study (Yang et al., 2023), they discuss the fine-tuning of the LLaMA-7B model on the MALLS dataset to achieve performance comparable to GPT-4, which they refer to as LOGICLLAMA. They note that fine-tuning for NL-FOL translation presents unique challenges beyond those of typical NLP tasks and address these with specific strategies.

LOGICLLAMA is capable of two types of tasks: direct translation of FOL from NL, and correction of FOL generated by more powerful models such as GPT-3.5. The latter, termed as the naive correction approach, involves a one-step correction process, leveraging the strengths of GPT-3.5 in capturing the essence of FOL rules and having a smaller model refine the output.

However, the direct translation (T1) and naive correction (T2) tasks, while easier to train, do not yield optimal results. The researchers found that employing a Chain-of-Thought (CoT) (Wei et al., 2022) correction technique, which involves producing intermediate steps during the correction, leads to better model performance .

The CoT correction process is divided into two types: SFT CoT correction (T3), which generates intermediate correction steps for synthetic data, and RLHF CoT Correction (T4), which corrects the actual outputs from GPT-3.5.

"If you steal from one author it's plagiarism; if you steal from many it's research." Wilson Mizner.

3.4.2 MALLS dataset

The MALLS dataset, an integral component of their study (Yang et al., 2023), was generated by collecting NL-FOL pairs from GPT-4, which was recognized as one of the most powerful LLMs available in 2023. The dataset comprises 34,000 pairs and is expected to grow in future iterations. The motivation for creating MALLS was to provide a resource for fine-tuning and evaluating NL-FOL translation models, offering an alternative to the costly and not fully public API access of GPT-4, and allowing for local deployment without privacy concerns.

Data collection from GPT-4 employed a dynamic prompting pipeline to ensure diversity and validity, including a frequency counter for n-grams, a prompter module to generate varied prompts, and a verifier to check FOL rule syntax using a context-free grammar approach. MALLS outnumbers and outshines both LogicNLI and FOLIO in diversity and contextual richness, with a significantly larger vocabulary size and more complex FOL rules.

As depicted in Tables 3.6 and 3.7, the MALLS dataset showcases a larger number of NL-FOL pairs, a broader vocabulary size, and greater complexity in FOL compared to other datasets. Figure 3.4 provides a visual representation of the dataset structure, offering insights into the string lengths and the complexity of the natural language and FOL examples.

Dataset	Source	#NL-FOL	**#Vocab**	#words	#literals
FOLIO	Expert	2K	5105	10.4	2.1
LogicNLI	Synthetic	12K	2061	13.9	2.8
MALLS	GPT-4	34K	22715	16.1	4.6

Table 3.6: General statistics of MALLS, LogicNLI, and FOLIO datasets. Source column indicates the way the data was obtained. NL-FOL counts the number of pairs in natural language an its FOL translation. Vocab shows the number of different words used along the whole dataset. words shows the average number of words for each sentence. Literals calculate the average number of either constants or variables used in the FOL translation

Dataset	$\forall$	$\exists$	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\oplus$
FOLIO	1111	182	421	631	167	1137	17	121
LogicNLI	2783	5327	10230	6590	2373	8712	3288	0
MALLS	32865	2036	4567	30143	6402	30667	3726	2150

Table 3.7: Logical operator statistics of MALLS, LogicNLI, and FOLIO datasets. Each number counts the number of instances for each logical symbol

The dataset exhibits high diversity in NL-FOL pairs, with a wide-ranging term vocabulary, suggesting a diverse distribution and the involvement of terms in semantically and contextually varied rules. While a simplistic verifier was implemented to check NL-FOL alignment by computing term frequency, the alignment check was not as rigorous as that in the FOLIO dataset creation due to constraints. As such, the dataset is recommended

"Our diversity is our strength. What a dull and pointless life it would be if everyone was the same." Angelina Jolie.

Figure 3.4: Visual representation of the dataset structure. The first row show the ditribution for the whole MALLS dataset for both Natural language sentences and FOL sentences. The three consecuent rows show examples of FOL and NL contained in the dataset

as “silver” labels for training, with “gold” label datasets used for evaluation. Despite this, a model trained solely on MALLS displayed comparable performance to GPT-4 when evaluated on gold standard datasets.

3.4.3 Evaluation and Results

The evaluation of the LOGICLLAMA model, according to the study, utilized two main tasks to gauge its performance in translating natural language to first-order logic: direct translation and Chain-of-Thought (CoT) correction. The model’s capability to directly translate and correct FOL rules generated by models such as GPT-3.5, alongside the reinforcement learning framework applied in the CoT correction task, underscore its potential in handling complex logical structures.

Performance was benchmarked against datasets such as FOLIO and LogicNLI, using the FOL BLEU and Logical Equivalence (LE) scores as metrics. CoT corrections were found to be particularly beneficial, especially in more challenging examples, with the number of CoT steps revealing that a few well-targeted corrections could lead to substantial performance enhancements, results for some GPT models and all the models developed by (Yang et al., 2023) can be found at Table 3.8.

Computing Logical Equivalence and BLEU Score

The LE score computation, a measure of similarity between two FOL rules R and R', proceeds through several steps after parsing the rules into a Context-Free Grammar (CFG) tree:

1. Identifying literals in both R and R', denoted as $P = [p_1, p_2, \ldots]$ and $Q = [q_1, q_2, \ldots]$.
2. Binding literals in P to those in Q or vice versa.
3. Generating truth tables for the bindings and calculating the overlap ratio.

Should parsing fail, indicating syntactic invalidity, the rule is scored zero.

“At the end of the day, you can’t control the results; you can only control your effort level and your focus.” Ben Zobrist.

FOL BLEU Score Computation

The FOL BLEU score is calculated using a specialized tokenizer tailored for the structure of first-order logic. This tokenizer is designed to split the formula into tokens corresponding to quantifiers, operators, and terms. The sequence of tokens generated by this tokenizer aligns with the leaf nodes of the CFG parse tree of the FOL formula. These nodes are listed in pre-order traversal, meaning that the nodes are visited in the order they are encountered, starting from the root and proceeding downwards, visiting all the nodes before proceeding to the siblings. An example of such a CFG parse tree is depicted in Figure 3.4.3 below.

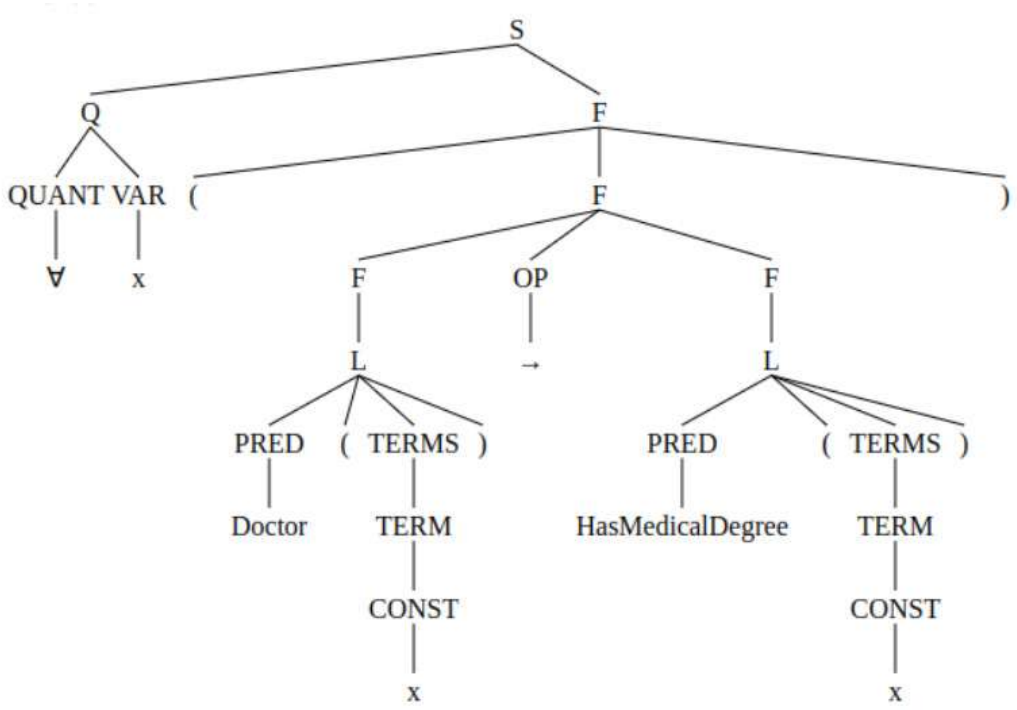

Figure 3.5: Context free grammar parse tree of FOL formula: $\forall x(\text{Doctor}(x) \rightarrow \text{HasMedicalDegree}(x))$.

Limitations of the LE Metric

I present that while the LE score may provide a high-level indication of model performance, it does not capture the effectiveness of the model in producing semantically equivalent translations. A high LE score does not necessarily mean that the translated formula is *almost equivalent* to the desired formula. The challenge of defining 'similarity' between two non-equivalent FOL formulas underlines the complexity of logic itself and suggests the need for more refined evaluation metrics.

"The only way to discover the limits of the possible is to go beyond them into the impossible." Arthur C. Clarke.

Methods	LogicNLI		FOLIO	
	FOL BLEU	FOL LE	FOL BLEU	FOL LE
GPT-3.5 0-shot	0.584	0.589	0.248	0.429
GPT-3.5 5-shot	0.905	0.918	0.341	0.767
GPT-4 0-shot	0.740	0.863	0.372	0.799
GPT-4 5-shot	0.913	0.989	0.400	0.855
Direct Translation	0.926	0.965	0.372	0.818
Naive Correction	0.934	0.970	0.373	0.840
SFT CoT Correction	0.663	0.830	0.332	0.730
RLHF CoT Correction	0.935	0.978	0.385	0.849

Table 3.8: Evaluation results on LogicNLI and FOLIO benchmarks. The table compares evaluations for four different models developed in (Yang et al., 2023) and other four models using either zero or few-shot prompting. LogicNLI is clearly easier for the models to perform well compare to FOLIO

3.4.4 Conclusion of their Work

This research presents LOGICLLAMA, a specialized language model (LM) uniquely developed for the task of translating natural language into first-order logic. The team has compiled and released an extensive dataset consisting of 34,000 sentence-level natural language to first-order logic (NL-FOL) pairs, sourced from GPT-4. This dataset has been instrumental in the fine-tuning process of LOGICLLAMA.

LOGICLLAMA, despite its relatively modest size of 7 billion parameters, achieves performance levels that are competitive with GPT-4 and surpasses GPT-3.5 in a comprehensive held-out NL-FOL benchmark. The effectiveness of LOGICLLAMA is largely due to the implementation of a novel training framework that integrates supervised fine-tuning (SFT) with reinforcement learning from human feedback (RLHF). This approach enables LOGICLLAMA to make incremental corrections, enhancing not only its outputs but also improving upon the translations provided by larger models, such as GPT-3.5. This capability represents a noteworthy advancement, indicating that smaller, specialized models can play a crucial role in executing complex translation tasks with high efficiency.

Final Remarks

In this Chapter we explored different approaches and models for translating NL into FOL and for classifying a set of premises and their conclusion as true, false or unknown. From rules based systems to harnessing the power of LLMs, we delve in details for three different previous works that represents three different approaches.

Our research aims to overpass in some of the metrics saw in this Chapter for accuracy, partial ratio and syntactic validity for the same test dataset as in FOLIO. Our approach includes fine-tuning the next version of Llama similarly to in MALLS and we will use both MALLS and FOLIO datasets for our training and testing. The following Chapter details the work done in our investigation.

"A conclusion is the place where you got tired thinking." Martin H. Fischer.

Chapter 4

Beyond Translation: Llama2 and Z3 in Logical Reasoning and NL Processing

This thesis introduces FOLARIS (First-Order Logic and Reasoning Interpretation System). This pioneering model represents a significant leap forward in translating natural language into first-order logic and facilitating automatic reasoning by creating a thinking fast and slow model. Llama2 will be used as the instuition a fast thinker of the model, in contrast Z3 will performed the slow thinking (this is a reference to the famous book of the braining functioning titled: *thinking, fast and slow* (Kahneman, 2011)). The overall approach of our work is summarized in Diagram 4.1.

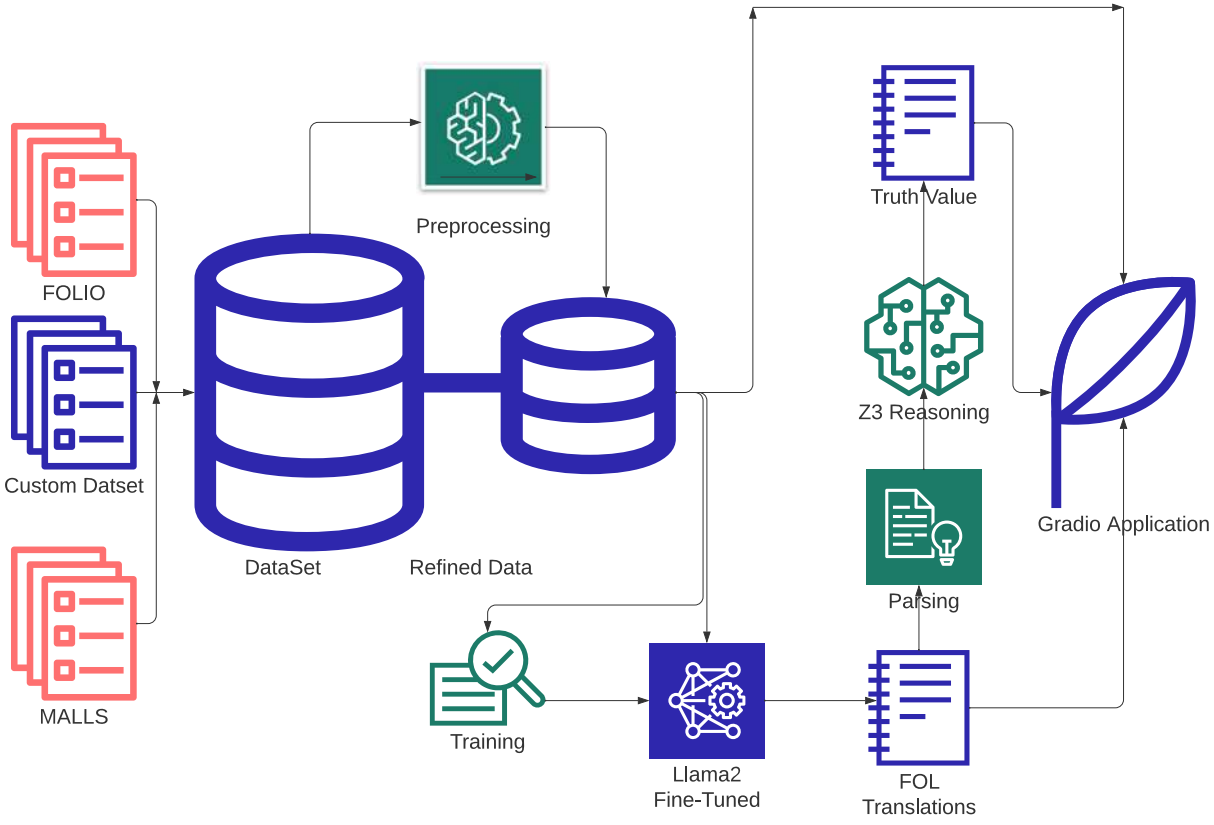

Figure 4.1: Overview of the thesis project

Overview of the Thesis Project

In Figure 4.1 Processes are marked in green, products produced by our work in blue, and products produced by a third party in orange.

We begin by creating a custom dataset and merging it with the FOLIO and MALLS datasets, as detailed in section 4.1.1. Next, we preprocess the data by filtering the merged dataset and structuring it according to the curriculum learning technique described in section 4.1.4. Following this, we split the refined data into test and train sets, which are then used to train Llama2 for translation tasks, see more in section 4.2.

At this stage, we leverage the fine-tuned Llama2 model to predict FOL formulas from the natural language sentences in the test dataset. Subsequently, these FOL formulas are parsed into the SMT-LIB format (see section 4.3.2). The parsed data is then passed to the Z3 reasoning process to determine the truth value, more details in section 4.3.2. Finally, we deploy the results using Gradio as discribed in section 4.4, displaying the original natural language premises, the translations, and the truth value.

4.1 Dataset Creation and Utilization

4.1.1 Merging Diverse Data Sources

My approach to fine-tuning the Llama 2 model involved the utilization of various datasets, each chosen for its unique characteristics and contributions to the training process. The datasets used include:

- **FOLIO Dataset**: Recognized for its quality, being expert-generated, the FOLIO dataset offers high accuracy and minimal errors. However, its relatively small size posed a limitation for effective fine-tuning of a large model like Llama 2.
- **MALLS Dataset**: This extensive dataset, comprising over 27,000 samples generated by GPT models, offers volume but with a trade-off in reliability. Particularly, longer samples in MALLS exhibited a higher propensity for errors.
- **Custom-Built Dataset**: To address specific shortcomings observed when training with only FOLIO and MALLS, I created a dataset targeting known issues, such as misidentification of constants and confusion in logical symbols. This dataset was constructed using lists of proper names, verbs, nouns, and adjectives to form atomic and compound FOL formulas, along with their natural language equivalents.

4.1.2 Custom Dataset Construction Technique

The artificial dataset was designed using a systematic approach with a limited vocabulary compounded by proper names and predicates where each predicate is either a verb, an adjective, or a noun. Atomic FOL formulas were created by randomly pairing a proper name with a predicate, followed by the construction of their natural language equivalents. Compound formulas were then formed using two atomic formulas and logical symbols like $\oplus$, $\vee$, $\leftrightarrow$, and $\neg$, corresponding to natural language expressions such as “either-or”, “or”, “if and only if”, and “not”. Table 4.1 shows the structure of the custom data. Due to

“Imagination is the beginning of creation. You imagine what you desire, you will what you imagine and at last you create what you will.” George Bernard Shaw.

the randomness when choosing two atomic formulas some of the statements do not make sense in coloquial language, for instance, "Mike is a conductor if and only if they are a constellation", also because of the robustness when generating the sentences some of them are grammarly incorrect, like "Cooper is either a actor or a trainer". Regardless of those minor mistakes, the idea is for the model to learn the "if and only if" phrases that correspond to the symbol $\leftrightarrow$, the difference between "either-or" and "or", and recognize when something is a constant and when a predicate; each of those misinterpretations by the model were encountered in previous experimentation as indicated in section 5.2.

NL	**FOL**
Lois is either melancholic or pessimistic.	Melancholic(lois) $\oplus$ Pessimistic(lois)
Cooper is either an actor or a trainer.	Actor(cooper) $\oplus$ Trainer(cooper)
Misael is a fruit if and only if they are a shape.	Fruit(misael) $\leftrightarrow$ Shape(misael)
Charlie is not an idea.	$\neg$Idea(charlie)
Mike is happy if and only if they are sleepy.	Happy(mike) $\leftrightarrow$ Sleepy(mike)

Table 4.1: This are five samples similiar to what the custom dataset contains. The samples aim to teach the model when to use biconditional, exclusive or, and negation correctly

4.1.3 Training Dataset Configuration

The final training dataset utilized to make FOLARIS comprised 1,496 samples from FOLIO, 9,782 samples from MALLS, and 2,000 samples from my custom dataset. The selection criteria included truncating formulas exceeding 74 characters to maintain quality and consistency. This resulted in a dataset exceeding 13,000 samples.

To align with the instruction-following nature of Llama, the dataset format was transformed to "FOL" and "instruction". Each input row in the "NL" column was prefixed with the instruction **"Construct a first-order logic expression based on the following sentence: "**, thereby guiding the model to generate the corresponding FOL expression.

This comprehensive dataset, combining expert-generated, GPT-generated, and custom-built samples, was integral to training the model effectively, addressing both volume and quality considerations while targeting specific translation and reasoning challenges in NL-FOL conversion.

4.1.4 Implementation of Curriculum Learning

Curriculum learning was a key strategy in training the model, aimed at gradually introducing the model to more complex aspects of the dataset. This approach was operationalized by adding hints to some training samples, thereby guiding the model's understanding of each component within the FOL formulas. The decision to include hints in a given sample followed a half-normal distribution, ensuring a gradual increase in complexity.

"A failure is not always a mistake, it may simply be the best one can do under the circumstances. The real mistake is to stop trying." B. F. Skinner.

The hints provided to the model included elements extracted from the FOL formula, such as symbols, predicate names, variables, and constants. The construction of the instruction for each sample varied depending on whether hints were included, as follows:

```
instruction = (
    "Construct a first-order logic expression based on the "
    "following sentence: " + row['NL']
)

if row['activation']:
    instruction += "\n\nHints:"
    instruction += "\n- Predicates to use: " +
    row['predicates'] + "."
    instruction += "\n- Variables within the predicates:
    " + row['variables'] + "."
    instruction += "\n- Constants within the predicates:
    " + row['constants'] + "."
    instruction += "\n- Logical symbols to use: " +
    row['symbols'] + "."
```

The activation of hints for a sample was determined by the probability $\exp(-i^2/(2\cdot\sigma^2))$ shown in Figure 4.2, where i is the index of the sample, N is the total number of samples, and $\sigma = N/5$. This distribution meant that approximately 25 percent of the samples included hints, with a higher concentration of hinted samples at the beginning of the training set. This structured approach allowed the model to initially learn from simpler examples and progressively advance to more complex ones, adhering to the philosophy of curriculum learning (Bengio, Louradour, Collobert, & Weston, 2009).

4.2 Fine-Tuning LLMs for NL-FOL Translation

The fine-tuning of Zephyr and especially Llama 2 model, which represents the latest iteration of the Llama series, focuses on translating individual sentences from natural language to first-order logic. For this crucial task, we utilize Ludwig (see appendex A), a code-free deep learning toolbox that simplifies the fine-tuning process. Ludwig's flexibility and user-friendly configuration system enable a streamlined and efficient fine-tuning operation. This approach is executed with the expectation that it will result in enhanced performance compared to the model's previous version.

To support the computationally intensive demands of training the Llama2 models and conducting the various experiments, including the development of the final iteration of FOLARIS, robust computational resources were utilized and they were provided by the faculty of Electrical Engeneering wihtin the University of Michoacan. A machine equipped with an NVIDIA A100 80GB PCIe GPU was leveraged, renowned for its formidable computational capabilities and expansive memory. This high-performance GPU was pivotal in efficiently training the models. The machine's specifications included:

"On two occasions I have been asked, 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question." Charles Babbage.

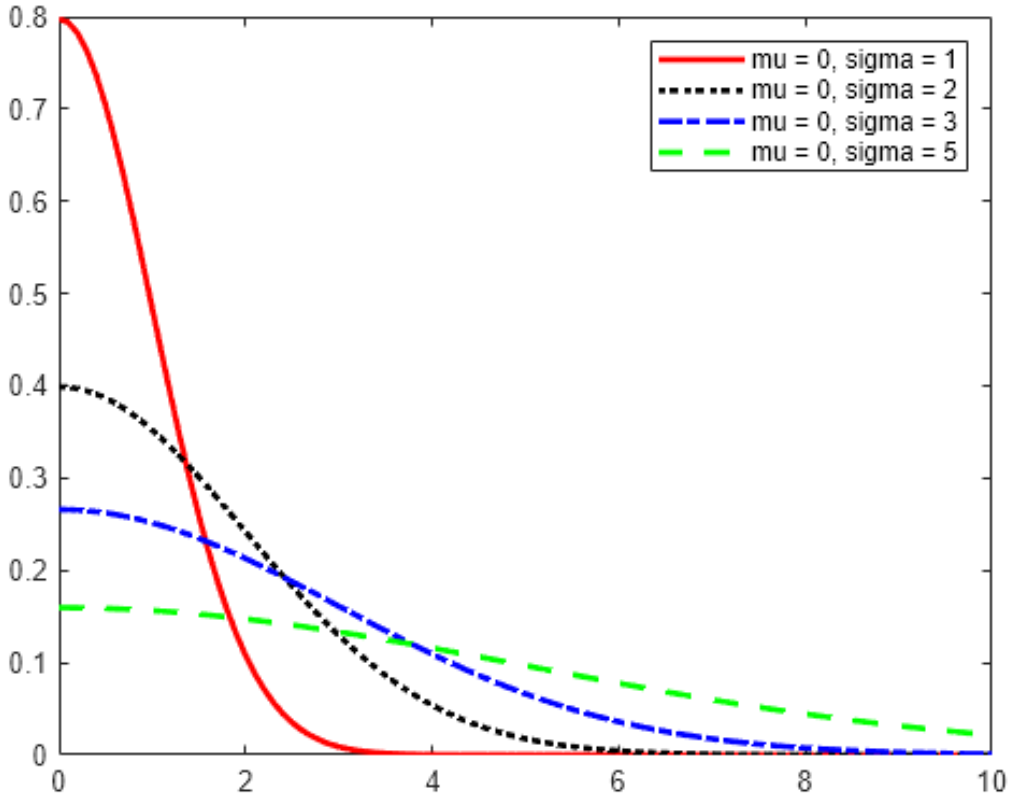

Figure 4.2: Half-normal distribution. All of the function shown has their centroid in 0 and they have different sigma values showing the differences in variance

- GPU Type: NVIDIA A100 80GB PCIe
- Total Memory: 81053MB
- Multi-Processor Count: 108
- CUDA Device Properties: Compute Capability 8.0

This configuration provided a variety of GPU compute capabilities to support a broad spectrum of CUDA architectures, which was essential for the sophisticated deep learning tasks at hand. The CUDA gencode flags specified for the machine were:

```
-gencode arch=compute_50,code=sm_50
-gencode arch=compute_60,code=sm_60
-gencode arch=compute_70,code=sm_70
-gencode arch=compute_75,code=sm_75
-gencode arch=compute_80,code=sm_80
-gencode arch=compute_86,code=sm_86
-gencode arch=compute_90,code=sm_90
```

"Be yourself. Create your own unique style." Kim Tae-hyung.

A single-node setup with one GPU was chosen for the experiments to ensure that the model training was conducted efficiently on a single machine. This setup eliminated the need for distributed computing across multiple nodes, thereby simplifying the development and fine-tuning process for FOLARIS.

4.2.1 Fine-Tuning Configuration for FOLARIS

The training for the main model was carried out with the following main characteristics:

- Adapter Type: LoRA
- Base Model: meta-llama/Llama-2-7b-hf
- Generation Parameters: max_new_tokens=256, temperature=0.01, top_k=8
- Input Features: instructions (text)
- Ludwig Version: 0.9.dev
- Model Type: llm
- Output Features: FOL (text)
- Preprocessing: global_max_sequence_length=512
- Training Split: 90% training, 5% validation, 5% test (random split)
- Prompt Template: instructions and response format
- Training Epochs: 5
- Learning Rate: 0.0002
- Optimizer: AdamW with weight decay=0.01

4.2.2 Rationale Behind Configuration Choices

We decided to train Llama2 using those specific configuration and hyperparameter based on experimentation and some guidance based on the results got by previous models trained. The following are some of the main reasons for choosing those hyperparameters and training configuration as well as the base model itself.

- Llama 2 was selected as the primary model for this research, being one of the best open-source large language models available. We specifically chose the 7 billion parameter variant to ensure compatibility with a single GPU setup, despite having hardware capable of supporting the 13 billion parameter model.
- Experiments were conducted using the Lora adapter, which was chosen for its potential to refine model performance. The rationale behind selecting Lora over other adapters and the specific advantages it offers for this research are discussed in detail in section 1.6.3.

"Is artificial intelligence less than our intelligence?" Spike Jonze.

- Regarding data distribution, 90% of the dataset was allocated for training, with the remaining 10% equally divided into 5% for validation and 5% for testing. This allocation strategy was chosen to maximize the training dataset while ensuring adequate samples for both validation and testing. The validation set was crucial for hyperparameter tuning, while the test set enabled an unbiased evaluation of the model's performance only for the loss function since we test in another dataset chosen from FOLIO. Although five epochs were generally optimal for minimizing loss and preventing overfitting, in some cases, fewer than five epochs sufficed to achieve the best performance, highlighting the need for vigilant monitoring to prevent overfitting.

- The prompt for the model was the instructional text discussed in the previous section. In configuring the model, a principle of simplicity was followed. Only essential elements were modified to suit our specific requirements:

  · **Temperature**: Set at a very low value to encourage determinism in the model's outputs. This is crucial for generating logical formulas where consistency and accuracy are paramount.

  · **Top-k**: Limited to the 8 most likely tokens to restrict the model's output to the most probable options, aiding in generating precise and relevant results.

  · **Learning rate**: A low learning rate was selected for fine-tuning to prevent the model from diverging significantly from its pre-trained knowledge. This careful adjustment was necessary to align the model with the new task without losing its foundational capabilities.

  · **Weight decay**: Additionally, weight decay was incorporated to counteract the tendency of the model to overfit, a challenge observed during initial experiments. The specific values for the learning rate and weight decay were determined through experimental tuning.

4.3 Reasoning Process

4.3.1 Z3 Solver

The culmination of this thesis is to ascertain whether a given set of premises and their conclusion (written in natural language) collectively constitute a tautology, a contradiction, or are contingent—paralleling the objectives of the work presented in (Han et al., 2022b). The novel aspect of our approach lies in leveraging the capabilities of the state-of-the-art reasoning tool, the Z3 Theorem Prover, commonly referred to as Z3. This powerful theorem solver, whose functionalities are detailed in appendex B, is adept at facilitating logical reasoning over FOL formulas.

One illustrative task is the verification of De Morgan's laws, which articulate the logical equivalence that the negation of a conjunction equates to the disjunction of the negations. Within the Z3 framework, we can represent this using functions to denote the

"In questions of science, the authority of a thousand is not worth the humble reasoning of a single individual." Galileo Galilei.

propositions a and b. The following Z3 script is configured to evaluate the equivalence $\neg(a \wedge b) \leftrightarrow (\neg a \vee \neg b)$:

```
(declare-fun a () Bool)
(declare-fun b () Bool)
(assert (not (= (not (and a b)) (or (not a) (not b)))))
(check-sat)
```

In this script, the `declare-fun` command declares a and b as Boolean functions. The script then asserts the negation of the equivalence between $\neg(a \wedge b)$ and $(\neg a \vee \neg b)$, with the `check-sat` command determining the satisfiability of this assertion.

Result:

```
unsat
```

An `unsat` (unsatisfiable) outcome implies that the negated proposition cannot hold true, thereby affirming the original proposition as a tautology and corroborating the logical equivalence posited by De Morgan's laws.

Z3 proves indispensable for deducing the truth values of a set of statements. However, a challenge arises since Z3 does not inherently process FOL or natural language. Instead, it requires translation into its native SMT-Lib format (which stands for Satisfiability Modulo Theories library). The subsequent sections will explore the conversion process from NL to SMT-Lib and the implications for our reasoning tasks.

4.3.2 From Natural Language to Logical Veracity

The reasoning process initiated in our study hinges on a sequence of methodical transformations, beginning with premises and a conclusion expressed in natural language. This initial phase is grounded in the realm of everyday language, laying a relatable foundation for the complex logical analysis that follows.

Transformation into Formal Logic (FOL)

Our Llama fine-tuned model is then used to translate these statements into FOL formulas. This step is crucial as it transforms natural language statements into a formal structure that can be processed logically.

Standardizing Predicates with Partial Ratio

To improve the consistency and precision of our logical analysis, we utilize the **partial ratio** metric explained in detail in section 5.1, which plays a crucial role in assessing the similarity of predicates within various premises. This approach is designed to tackle a frequent issue encountered with outputs from language models, where predicates that differ slightly in phrasing may nonetheless share identical meanings. Advanced language models, such as Llama2, may produce a range of responses, resulting in predicates that are akin but not precisely the same. We can detect and consolidate these akin predicates into a standardized format by implementing the partial ratio metric. Such standardization

"Hope is the only universal liar who never loses his reputation for veracity." Robert Green Ingersoll.

is vital because Z3, the solver we use for analysis, interprets predicates with different phrasings as separate entities, which could distort the logical interpretation. By ensuring that semantically similar predicates are treated as equivalent, we enhance the accuracy and reliability of the logical processing in Z3.

Tree Structure Utilization

Leveraging the tree structure of FOL formulas, as conceptualized by (Yang et al., 2023), each formula derived from the model is meticulously analyzed. This tree structure proves instrumental in systematically deconstructing the formulas, facilitating the extraction and precise placement of various logical elements such as quantifiers, constants, variables, predicates, and logical symbols.

The implementation of the tree structure used in this analysis was directly adapted from the work of . The specific program, available in their GitHub repository (Han et al., 2022a), represents FOL formulas as tree structures.

To exemplify the application of this approach, consider the natural language statement: "If Rockie is not both a turtle and a squirrel, then Rockie is either cute or skittish". This statement, when translated into a logical formula, is represented as $\neg(Turtle(rockie) \wedge Squirrel(rockie)) \rightarrow (Cute(rockie) \oplus Skittish(rockie))$. Utilizing the tree structure program, this formula can be deconstructed not only to visually display the relationships between the various logical components, as shown in Figure 4.3, but also to create a local representation in Python that differs from a string. This becomes important when converting formulas into the SMT-Lib format.

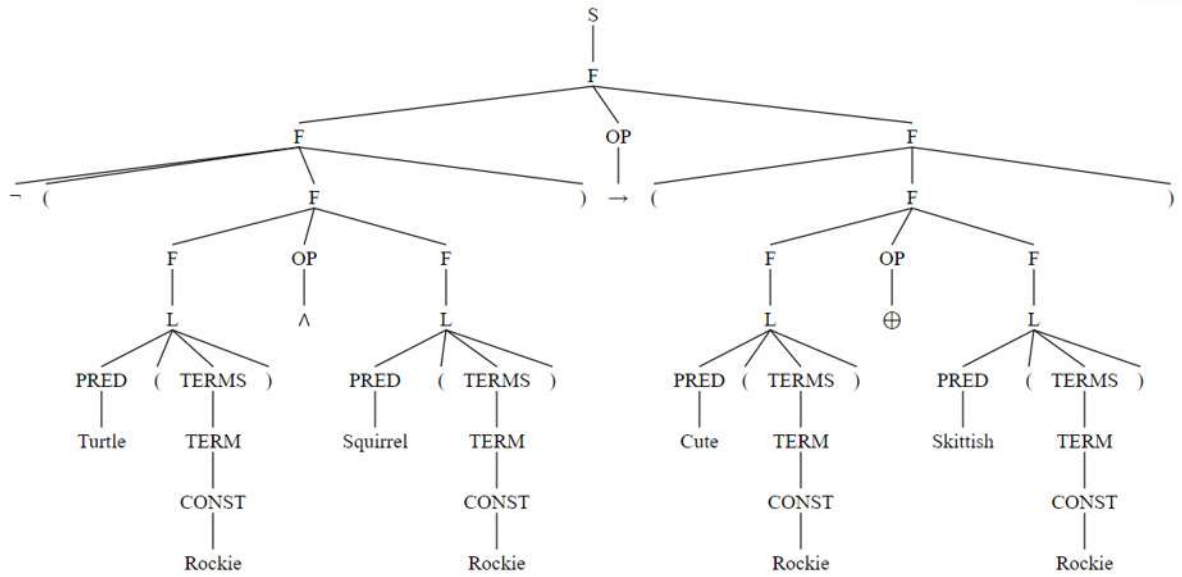

Figure 4.3: Tree Structure Example for the formula $\neg(Turtle(rockie) \wedge Squirrel(rockie)) \rightarrow (Cute(rockie) \oplus Skittish(rockie))$. This structure allow us to use code for converting to SMT-LIB easily

"Some people call this artificial intelligence, but the reality is this technology will enhance us. So instead of artificial intelligence, I think we'll augment our intelligence." Ginni Rometty.

Conversion to SMT-LIB Format

The conversion of FOL formulas to SMT-LIB format is a critical step in our process, realized through a Python program I developed, available at our repository (Pelayo, 2023d). This program employs several key functions to systematically transform the tree-structured FOL formulas into a format suitable for the Z3 solver. The algorithm's workflow can be summarized as follows:

- Formula Reconstruction: The central function, `formula`, recursively processes the nodes of the FOL tree structure. It interprets and rebuilds the formula, translating FOL constructs into their SMT-LIB counterparts. This includes handling logical operators, quantifiers, predicates, and terms, ensuring that each element is accurately represented in the target format.
- Extracting Variables: The `extract_variables` function parses the FOL string to identify and extract quantified variables. This step is crucial for understanding the scope of each variable within the formula.
- Extracting Constants: The process involves examining each term in the FOL formula's tree structure. If a term is not found in the set of variables (previously identified by the `extract_variables` function), it is classified as a constant. This approach ensures that all constants are accurately identified and distinguished from variables. These constants are then declared appropriately in the SMT-Lib format, which is crucial for correctly interpreting and evaluating the formula in the Z3 solver.
- Determining Predicate Arity: The `CountArity` function calculates the arity of predicates within the formula. This is essential for the correct declaration of the predicates.
- SMT-Lib Declarations: The `generate_smtlib_declarations` function generates necessary SMT-Lib declarations for predicates and constants found in the formula, setting the stage for their logical evaluation.
- Full Conversion Process: The `fol_tree_to_smtlib` function combines these steps, converting the entire FOL tree into an SMT-Lib compatible string. This includes handling of quantifiers, logical operations, and structuring of the formula according to SMT-Lib syntax.

Through this structured and methodical approach, the program effectively translates complex FOL formulas into a format amenable to logical processing by the Z3 solver, playing a pivotal role in the overall reasoning process.

Consider the logical formula derived from the natural language statement as an illustrative example: "If Rockie is not both a turtle and a squirrel, then Rockie is either cute or skittish." This statement is formalized as $\neg(\text{Turtle}(\text{rockie}) \wedge \text{Squirrel}(\text{rockie})) \rightarrow (\text{Cute}(\text{rockie}) \oplus \text{Skittish}(\text{rockie}))$.

When processed through the developed Python program, this FOL formula is translated into the following SMT-LIB string:

"All of our reasoning ends in surrender to feeling." Blaise Pascal.

```
(declare-fun Turtles (Int) Bool)
(declare-fun Squirrel (Int) Bool)
(declare-fun Cute (Int) Bool)
(declare-fun Skittish (Int) Bool)
(declare-const Rockie Int)
(assert (=> (not (and (Turtles Rockie)
(Squirrel Rockie))) (xor (Cute Rockie)
(Skittish Rockie))))
```

This string precisely defines the logical constructs and relationships present in the original FOL formula. Each predicate and constant is declared with its respective type, ensuring that the Z3 solver accurately interprets the formula's logical structure.

To integrate this string into the Z3 solver, we employ the function `parse_smt2_string`. This function converts the SMT-Lib string into a format that is directly compatible with the Z3 solver. For the given example, the `parse_smt2_string` function transforms the string into the following SMT-LIB format:

```
[Implies(Not(And(Turtles(Rockie),
Squirrel(Rockie))), Xor(Cute(Rockie),
Skittish(Rockie)))]
```

Logical Evaluation with Z3

First if the label is true we must be able to proof the conclusion based on solely on the premises. To do so we proceed by reduction to absurdity. All premises, coupled with the negation of the conclusion are transformed by the process described below and added within the Z3 solver. The solver automatically evaluate the satisfiability of the set premises and the negated conclusion, if this combination is deemed unsatisfiable we have found a contradiction in the set of sentences and therefore the premises implie the conclusion.

It can be the case that the previous process does not return unsatisfiable, in that case we proceed as follows: first we retract the negation of the conclusion from the solver to get back where we only have stored the premises. We will try to proof that the negation of the conclusion is true by reduction to absurdity. So we assumed the conclusion true which means in practical terms to add it to the solver. Should the solver find this revised set unsatisfiable, it indicates the conclusion, along with the premises are logically contradictory and then the we have proof the conclusion is false. Finally in scenarios where the solver validates both sets as satisfiable, it means the conclusion couldn't be proven as false nor true and thus we set as unknown (it actually means that the conlcusion is undecidable from the premises, in the mathematical sense. But in this case it can also mean that Z3 was not able to find the proof)

4.4 Model Deployment Using Gradio

To enhance the accessibility and facilitate the practical use of FOLARIS, the model has been deployed using Gradio—an open-source Python library that creates a user-friendly

"An algorithm must be seen to be believed." Donald Knuth.

interface for machine learning models. Gradio interfaces are designed to be intuitive, allowing users to easily interact with the underlying model.

Figure 4.4: Initial view of the Gradio interface for FOLARIS. In the left we a box where we type

Figure 4.4 displays the initial view of the Gradio interface for FOLARIS. The interface is bifurcated into two primary sections: on the left is the "Natural Language" input field where users are invited to input natural language sentences; on the right are the "First-Order Logic" and "SMT-LIB Declarations" output fields, where the translations and truth declarations produced by FOLARIS are exhibited.

In the "Natural Language" input field, users should enter their premises, ensuring each is devoid of periods, as the "." character functions as a delimiter to separate multiple premises. The final premise entered is treated as the conclusion in a set of premises; however, if only a single premise is provided, it is regarded as a standalone translation rather than part of an argumentative structure. After inputting the premises, users can initiate the translation process by clicking the "Submit" button.

Upon processing the input, the "First-Order Logic" output field presents the resulting translations, while the "SMT-LIB Declarations" field displays corresponding declarations for inputs comprising at least two sentences. If only a single sentence is input, the "SMT-LIB Declarations" field will indicate "No Conclusion" to reflect the absence of an argumentative structure. These outputs afford users the opportunity to verify the logical statements formulated by FOLARIS. Additionally, the interface incorporates a "Flag" button, permitting users to signal any discrepancies or errors in the translations, thereby contributing to the continuous refinement of the model. This feature is particularly instrumental in assembling a dataset of unlearned patterns, which can be leveraged for future model fine-tuning. This interactive setup through Gradio not only democratizes access to advanced natural language processing tools like FOLARIS but also fosters an environment conducive to real-time feedback and collaborative enhancement of the model's capabilities.

With the creation of FOLARIS complete, it's essential to provide access to the model and associated resources. FOLARIS can be loaded through the Hugging Face platform using the following Python code:

```
from peft import PeftModel, PeftConfig
from transformers import AutoModelForCausalLM

config = PeftConfig.from_pretrained("Fermat111/FOLARIS")
model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-7b-hf")
```

"Unity is strength... when there is teamwork and collaboration, wonderful things can be achieved." Mattie Stepanek.

```
model = PeftModel.from_pretrained(model, "Fermat111/FOLARIS")
```

Versions of the model with quantization are available at (Pelayo, 2023a). For practical examples, including loading the model and deploying it with Gradio within Colab, users can find the colab notebook at (Pelayo, 2023c)

It is noteworthy that loading the model may require a premium Colab account due to the limitations of common accounts, which only allow for 12.67GB of RAM and 78.19GB of disk space. The Colab notebook includes code that retrieves my Git project, which is necessary for running certain functions. Interested users can inspect the code, including the validity, Tree2SMT_LIB, and FOL2tree Python scripts, at the following GitHub repository: (Pelayo, 2023d) Furthermore, the datasets utilized for training and testing, which are detailed in sections 4.1 and 5.2, can be spotted and loaded directly through Hugging Face with Python:

```
from datasets import load_dataset

dataset = load_dataset("Fermat111/FOL")
train = dataset['train']
test = dataset['test']
```

These resources ensure that users can not only utilize FOLARIS for their own applications but can also contribute to its ongoing development and refinement.

Final Remarks

We have embarked on a journey to create FOLARIS (First-Order Logic and Reasoning Interpretation System). This model is capable of translating natural language into first-order logic and automating logical reasoning. By combining the strengths of Llama2 as the intuitive, fast-thinking component and Z3 as the meticulous, slow-thinking element.

Throughout this chapter, we detailed the approach taken to develop FOLARIS. We began by curating a diverse dataset, merging expert-generated, GPT-generated, and custom-built samples. The dataset was structured and preprocessed, employing curriculum learning techniques.

Fine-tuning Llama2 was achieved with precision, leveraging Ludwig's user-friendly interface and robust computational resources. The rationale behind our configuration choices was explained.

The reasoning process was augmented by the powerful Z3 solver, which facilitated the logical evaluation of FOL formulas. We navigated the challenge of translating natural language into SMT-LIB format, ensuring compatibility with Z3's reasoning capabilities. This transformation was accomplished through a systematic conversion process, using tree structures and our own algorithms.

Finally, we deployed FOLARIS using Gradio, making it accessible to users through an intuitive interface. In the following chapter we will analyse the results obtained by FOLARIS and other experiments done

"Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program."Linus Torvalds.

Chapter 5

From Theory to Practice: Analyzing the Results of NL-to-FOL Conversion

This chapter unfolds the comprehensive evaluation conducted through a series of 49 experiments, which were instrumental in fine-tuning and assessing the efficacy of models built on the Llama 2 architecture, alongside a number of experiments with the Zephyr model. The fine-tuning procedure was judiciously engineered to ensure model stability both with and without the application of quantization techniques. The strategic application of quantization was aimed at condensing the model's footprint to enhance deployment efficiency, while still preserving the integrity of its performance and stability.

Particular attention is given to six models that stand as the cornerstone of this research. These models are elaborated in Section 5.2, setting the stage for a detailed performance analysis. Their evaluation is grounded on the metrics delineated in Section 5.1, providing a multifaceted view of their capabilities in translating natural language into first-order logic. This focused approach allows for a deeper understanding of the nuances in model performance, particularly in the context of the quantization's influence on the fine-tuning process.

5.1 Metrics

Assessing the performance of our models, we employed a set of metrics designed to capture different dimensions of translation quality and logical reasoning. These metrics are vital for understanding the strengths and limitations of the models under investigation.

Syntactic Validity (SynV): This binary metric determines whether the output formula is a valid FOL expression. We measure the percentage of syntactically valid formulas out of the total number of testing samples. Validation is conducted using the FOL2Tree parser; if it generates a parse tree without errors, we consider the formula syntactically valid.

Fuzzy Match Score: This score compares the similarity of predicate strings in the model-generated formulas to the reference formulas. The metric is quantified using the

"The single best machine to measure trust is a human being. We haven't figured out a metric that works better than our own sort of, like, 'There's something fishy about you.' " Simon Sinek.

fuzz.partial_ratio function from Python's FuzzyWuzzy library. Unlike the standard Levenshtein distance, which considers the entire string, the Fuzz Partial Ratio focuses on the longest common substring, accommodating the model's tendency to generate predicates that may contain additional but semantically consistent words. This approach is more lenient towards variations in predicate expression that still preserve meaning.

Accuracy: We define accuracy as the percentage of arguments for which the model, in conjunction with the Z3 solver, correctly determines the truth value. This is compared to the predefined label of each set of premises and conclusion, providing a direct measure of the model's reasoning capability.

Qualitative Analysis: Beyond these quantifiable metrics, we conducted a qualitative analysis of selected translations. This examination focused on the model's ability to correctly translate fundamental symbols, recognize constants, differentiate between predicates of varying arity, and appropriately use quantifiers. This analysis helps to illustrate the model's nuanced understanding of logical expressions.

These metrics together provide a comprehensive picture of the models' performance, from basic syntactical correctness to more complex aspects of logical reasoning and translation fidelity.

5.2 Overview of Key Experiments

Fine-tuning efficacy of the Llama 2 and Zephyr models was meticulously examined through experiments utilizing a hybrid dataset comprised of FOLIO, MALLS, and a custom dataset detailed in section 4.1.2. This dataset was partitioned into 90% for training, with the remainder equally split between testing and validation. The following discussion presents key experiments and their outcomes, with a focus on the performance analysis for each metric used.

For assessing the best loss value, we utilized the test and validation datasets, which mirror the structure of the training set. When examining accuracy and syntactic validity, we employed a subset of an earlier version of FOLIO, available at (Pelayo, 2023b). This subset can be loaded using:

```
from datasets import load_dataset
dataset = load_dataset("Fermat111/FOL")
test = dataset['test']
```

It comprises 108 samples, each consisting of a set of premises in both NL and FOL, along with a conclusion in NL and an accompanying label indicating the truth value (true, false, or unknown). Only the conclusion is processed by the model to generate the corresponding FOL formula. The formula's syntactic correctness is verified (Syntactic Validity), and the procedure outlined in chapter 4 is applied to ascertain the truth value, which is then compared to the labeled truth value.

The Fuzzy Match score was evaluated using the same dataset as accuracy, but here the focus was on the predicates within the premises translated to FOL. The model-generated predicates are compared with the reference FOL predicates to calculate the score.

Beyond quantifiable metrics, we conducted a qualitative analysis of selected translations to evaluate the model's proficiency in accurately translating fundamental symbols,

"All life is an experiment. The more experiments you make the better. " Ralph Waldo Emerson.

recognizing constants, differentiating between predicates of varying arity, and using quantifiers appropriately. This examination illuminates the model's nuanced understanding of logical expressions, providing insights into its ability to navigate complex logical constructs.

For this analysis, from the sentences below we selected the first ten premises for translation into FOL, with the eleventh serving as the conclusion. The translations were generated by ChatGPT 4, and also we asked GPT about the truth value answering correctly demonstrating ChatGPT4's capability to reason and to translate from NL into FOL is still at another level compared with even our model.

1. $\forall x\,(\text{Student}(x) \rightarrow (\text{Studies}(x, \text{Mathematics}) \vee \text{Studies}(x, \text{Physics})))$
   - Every student either studies Mathematics or Physics, where the intended exclusive 'or' was not accurately captured, suggesting a misunderstanding of the logical operator.

2. $\forall x\,((\text{Student}(x) \wedge \text{Studies}(x, \text{Mathematics}) \wedge \text{Attends}(x, \text{LogicClass})) \rightarrow \text{Passes}(x, \text{LogicExam}))$
   - This indicates an accurate translation, properly identifying the conditional relationship between studying, attending classes, and passing exams.

3. $\forall x\,(\text{MathematicsProfessor}(x) \rightarrow \text{Researcher}(x))$
   - If someone is a Mathematics professor, then they are also a researcher, correctly employing the implication logical operator.

4. $(\text{Studies}(\text{Jose}, \text{ComputerScience}) \leftrightarrow \text{Studies}(\text{Jose}, \text{Biology}))$
   - Jose studies Computer Science if and only if he studies Biology, showcasing the model's ability to handle biconditional expressions, though the exclusive nature of 'either-or' remains a challenge.

5. $\neg\text{Studies}(\text{Jaime}, \text{Chemistry})$
   - Jaime does not study Chemistry, effectively translating the negation.

6. $\exists x\,(\text{Professor}(x) \wedge \text{Teaches}(x, \text{Mathematics}) \wedge \text{Teaches}(x, \text{Physics}))$
   - There is a professor who teaches both Mathematics and Physics, accurately using the existential quantifier to denote the existence of such a professor.

7. $\text{Student}(\text{Oscar}) \wedge ((\text{Won}(\text{Oscar}, \text{MathCompetition}) \vee \text{Won}(\text{Oscar}, \text{PhysicsCompetition})) \wedge \neg\,(\text{Won}(\text{Oscar}, \text{MathCompetition}) \wedge \text{Won}(\text{Oscar}, \text{PhysicsCompetition}))$
   - Oscar is a student who has won a Mathematics competition or a Physics competition but not both, demonstrating an improved handling of the exclusive 'or'.

8. $\forall x\,(\text{Teaches}(x, \text{Physics}) \rightarrow \text{Genius}(x))$
   - Someone who teaches Physics is considered a Genius, correctly translating the implication.

"Great innovation only happens when people aren't afraid to do things differently." Georg Cantor.

9. $\forall x \forall y\,(\text{Studies}(x, y) \rightarrow \text{Student}(x))$
   - If someone studies anything, then they are students, effectively using universal quantifiers.

10. $\forall x\,(\text{Passes}(x, \text{LogicExam}) \rightarrow \text{Awarded}(x, \text{Medal}))$
    - Someone who passes the Logic exam is awarded a Medal, accurately employing the implication operator.

11. $\exists x\,(\text{Genius}(x) \wedge \text{Researcher}(x))$
    - Someone is a Genius and also a Researcher, correctly using the existential quantifier.

We performed more than 50 experiments to get the best possible model, which was experiment 35. In the following paragraphs, we will delve into comparisons of experiment 21, 22, 27, 34, 35, 40 and 49 that are in the opinion of the author the most important that exemplified all the experimentation process.

Experiment 21

Llama-2 7b was trained using LoRA without quantization on a data set combining approximately 6k FOLIO samples (including duplicates) and approximately 10k MALLS samples. Curriculum learning was applied. The best loss recorded for it was 0.1422, with an accuracy of 67.59%.

Some of the conclusions from the results when translating the 11 sentences above are the following ones: the model generally creates predicate names that accurately capture the essence of the natural language statements. The model correctly identifies proper names and distinguishes between constants and predicates in some instances, but not consistently, as evidenced by the translation of *"Jose studies Computer Science if and only if he studies Biology"* into "$\forall x(studiesbiology(x) \rightarrow studiesComputerScience(x))$". Here, the model fails to include "Jose" and incorrectly interprets "if and only if" biconditional as a simple conditional.

The model often opts for predicates with a lower arity than necessary, resulting in overly complex predicates names. For example, "$TeachesMathematics(x)$" was one of the results that could be more concisely translated as "$Teaches(x, \text{mathematics})$". Although not incorrect per se, as both convey the same meaning, it is preferable to use the latter form for reasons discussed in chapter 2, such as when teaching involves multiple subjects, and mathematics could be treated as a constant.

Quantifiers are generally managed correctly by this model, except in the translation of *"Someone who teaches Physics is considered a Genius"* where the model produces "$\exists x(teachesphysics(x) \rightarrow genius(x))$". This indicates a misunderstanding of the quantifier's scope in relation to the original statement.

The model also struggles with differentiating between inclusive and exclusive "or", particularly in premises 1 and 7, where exclusive "or" would have been the correct interpretation. However, this model reliably uses the symbols for "conditional", "and", "inclusive or", and "for all" when they are appropriate. This consistency suggests that, while the model recognizes when these logical operators are required, it does not always discern when they are not, which can lead to their erroneous inclusion in the translated formula.

These are the whole translations for the 11 sentences:

"I love fools' experiments. I am always making them. " Charles Darwin.

1. $\forall x\,(\text{studymathematics}(x) \vee \text{studyphysics}(x))$
2. $\forall x\,(\text{student}(x) \wedge \text{studymathematics}(x) \wedge \text{attendlogicclass}(x) \rightarrow \text{passlogicexam}(x))$
3. $\forall x\,(\text{mathematicsprofessor}(x) \rightarrow \text{researcher}(x))$
4. $\forall x\,(\text{jose}(x) \leftrightarrow (\text{studycomputerscience}(x) \leftrightarrow \text{studybiology}(x)))$
5. $\neg\text{studychemistry}(\text{jaime})$
6. $\exists x\,(\text{professor}(x) \wedge \text{teachesmathematics}(x) \wedge \text{teachesphysics}(x))$
7. $\text{student}(\text{oscar}) \wedge (\text{won}(\text{oscar}, \text{mathematicscompetition}) \vee \text{won}(\text{oscar}, \text{physicscompetition})) \wedge \neg\,(\text{won}(\text{oscar}, \text{mathematicscompetition}) \wedge \text{won}(\text{oscar}, \text{physicscompetition}))$
8. $\forall x\,(\text{studyphysics}(x) \rightarrow \text{genius}(x))$
9. $\forall x\,(\text{study}(x) \rightarrow \text{student}(x))$
10. $\forall x\,(\text{passlogicexam}(x) \rightarrow \text{awardedmedal}(x))$
11. $\exists x\,(\text{genius}(x) \wedge \text{researcher}(x))$

Experiment 22

Zephyr-7b-alpha underwent 4-bit quantization training exclusively on FOLIO, applying curriculum learning. The best loss for this model was 0.1068 with a 56.48% accuracy.

In contrast to Experiment 21, this model accurately handles "exclusive or" and the biconditional expressions. However, it exhibits increased inaccuracies in the use of quantifiers. Specifically, the model tends to erroneously interpret "some-" as signaling an existential quantifier, as demonstrated in the examples: *"Someone who passes the Logic exam is awarded a Medal"* translated to "$\exists x(logicexam(x) \rightarrow medal(x))$", and *"Someone who teaches physics is considered a genius"* rendered as "$\exists x(teachesphysics(x) \rightarrow genius(x))$". This inclination toward existential quantification regardless of context marks a significant area for improvement.

Additionally, the model continues to struggle with recognizing constants, not only in the previously mentioned case involving "Jose" but also in translating *"Oscar is a student who has won a Mathematics competition or a Physics competition but not both"* into $\exists x(student(x) \wedge wonmathematicscompetition(x) \oplus wonphysicscompetition(x))$. This translation further highlights the model's difficulty in accurately representing unique individuals within logical expressions.

While some improvements in predicate arity were noted, indicating a better grasp of the appropriate complexity for certain logical relationships, other instances remain

"I think that in the discussion of natural problems we ought to begin not with the Scriptures, but with experiments, and demonstrations. " Galileo Galilei.

unchanged from Experiment 21. This inconsistency in applying the correct arity suggests that while the model has made strides in understanding certain logical constructs, it still lacks a consistent approach to translating natural language into first-order logic with precision.

Here there are the FOL formulas generated by this model:

1. $\forall x\,(\text{student}(x) \rightarrow (\text{studiesmathematics}(x) \vee \text{studiesphysics}(x)))$
2. $\forall x\,(\text{student}(x) \wedge \text{studiesmathematics}(x) \wedge \text{attendslogicclass}(x) \rightarrow \text{passeslogicexam}(x))$
3. $\forall x\,(\text{mathematicsprofessor}(x) \rightarrow \text{researcher}(x))$
4. $\forall x\,(\text{jose}(x) \leftrightarrow (\text{studiescomputerscience}(x) \leftrightarrow \text{studiesbiology}(x)))$
5. $\neg\text{studies}(\text{jaime}, \text{chemistry})$
6. $\exists x\,(\text{professor}(x) \wedge \text{teaches}(x, \text{mathematics}) \wedge \text{teaches}(x, \text{physics}))$
7. $\text{student}(\text{oscar}) \wedge (\text{won}(\text{oscar}, \text{mathematicscompetition}) \vee \text{won}(\text{oscar}, \text{physicscompetition})) \wedge \neg\,(\text{won}(\text{oscar}, \text{mathematicscompetition}) \wedge \text{won}(\text{oscar}, \text{physicscompetition}))$
8. $\forall x\,(\text{teachesphysics}(x) \rightarrow \text{genius}(x))$
9. $\forall x\,(\text{study}(x) \rightarrow \text{student}(x))$
10. $\forall x\,(\text{passeslogicexam}(x) \rightarrow \text{awardedmedal}(x))$
11. $\exists x\,(\text{genius}(x) \wedge \text{researcher}(x))$

Experiment 27

Utilizing only the MALLS dataset (28k samples) and 4-bit quantization, Zephyr-7b-alpha achieved a best loss of 0.1063 and 53.70% accuracy. For the translations of 27 we observe that the model successfully translates the sentence *"Oscar is a student who has won a Mathematics competition or a Physics competition but not both"*, correctly capturing the exclusive "or" condition. However, the model still encounters challenges with "either-or" expressions, often translating them as a standard "or", which dilutes the intended exclusivity of these statements.

Regarding the management of predicate arity, Experiment 27 demonstrates a regression to performances similar to Experiment 21, with the quality of translations not meeting the improved standards observed in Experiment 22. This inconsistency in handling predicate arity suggests that while the model can accurately represent certain logical

"A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas. " G.H. Hardy.

constructs, it struggles to maintain a uniform approach across different logical expressions, particularly in distinguishing between more nuanced logical operators and efficiently structuring predicates. The entire set of predictions is:

1. $\forall x\,(\text{student}(x) \rightarrow (\text{studiesmathematics}(x) \vee \text{studiesphysics}(x)))$
2. $\forall x\,(\text{student}(x) \wedge \text{studiesmathematics}(x) \wedge \text{attendslogicclass}(x) \rightarrow \text{passeslogicexam}(x))$
3. $\forall x\,(\text{mathematicsprofessor}(x) \rightarrow \text{researcher}(x))$
4. $\forall x\,(\text{jose}(x) \leftrightarrow (\text{studiescomputerscience}(x) \leftrightarrow \text{studiesbiology}(x)))$
5. $\neg\text{studieschemistry}(\text{jaime})$
6. $\exists x\,(\text{professor}(x) \wedge \text{teachesmathematics}(x) \wedge \text{teachesphysics}(x))$
7. $\text{student}(\text{oscar}) \wedge (\text{wonphysicscompetition}(\text{oscar}) \vee \text{wonphysicscompetition}(\text{oscar})) \wedge \neg\,(\text{wonphysicscompetition}(\text{oscar}) \wedge \text{wonphysicscompetition}(\text{oscar}))$
8. $\forall x\,(\text{teachesphysics}(x) \rightarrow \text{genius}(x))$
9. $\forall x\,(\text{study}(x) \rightarrow \text{student}(x))$
10. $\forall x\,(\text{passeslogicexam}(x) \rightarrow \text{awardedmedal}(x))$
11. $\exists x\,(\text{genius}(x) \wedge \text{researcher}(x))$

Experiments 34

Llama-2 7b was trained on the combined dataset without curriculum learning and achieved a loss of 0.2091 and 67.59% accuracy. Experiment 34 reveals mixed outcomes in the model's logical interpretation capabilities. While it regresses in accurately processing "either-or" expressions, translating them as a generic "or", a significant improvement is observed in the recognition of constants. For the first time in our series of experiments, this model correctly identifies proper names within statements, marking a notable advancement in handling specific entities.

Furthermore, the model adopts a more structured approach to naming predicates, consistently starting every predicate with an uppercase letter and employing uppercase letters to distinguish between words within predicate names, as exemplified by "$\forall x(TeachesPhysics(x) \rightarrow Genius(x))$". These enhancements in recognizing constants and structuring predicate names significantly contribute to the clarity and readability of the logical expressions generated by the model.

"Mathematics knows no races or geographic boundaries; for mathematics, the cultural world is one country. " David Hilbert.

The improvements observed in Experiment 34, particularly in the areas of constant recognition and predicate naming conventions, can be attributed to the bespoke dataset constructed for this purpose. This dataset's design evidently played a pivotal role in training the model to better grasp the nuances of logical expression, showcasing the direct impact of tailored dataset composition on model performance. these are the whole results:

1. $\forall x(\text{student}(x) \rightarrow (\text{studiesmathematics}(x) \vee \text{studiesphysics}(x)))$
2. $\forall x(\text{student}(x) \wedge \text{studiesmathematics}(x) \wedge \text{attendslogicclass}(x) \rightarrow \text{passeslogicexam}(x))$
3. $\forall x(\text{mathematicsprofessor}(x) \rightarrow \text{researcher}(x))$
4. $\forall x(\text{jose}(x) \leftrightarrow (\text{studiescomputerscience}(x) \leftrightarrow \text{studiesbiology}(x)))$
5. $\neg\text{studieschemistry}(\text{jaime})$
6. $\exists x(\text{professor}(x) \wedge \text{teachesmathematics}(x) \wedge \text{teachesphysics}(x))$
7. $\text{student}(\text{oscar}) \wedge (\text{wonphysicscompetition}(\text{oscar}) \vee \text{wonphysicscompetition}(\text{oscar})) \wedge \neg(\text{wonphysicscompetition}(\text{oscar}) \wedge \text{wonphysicscompetition}(\text{oscar}))$
8. $\forall x(\text{teachesphysics}(x) \rightarrow \text{genius}(x))$
9. $\forall x(\text{study}(x) \rightarrow \text{student}(x))$
10. $\forall x(\text{passeslogicexam}(x) \rightarrow \text{awardedmedal}(x))$
11. $\exists x(\text{genius}(x) \wedge \text{researcher}(x))$

Experiments 35

Experiment 35, conducted similarly to Experiment 34 but with the incorporation of curriculum learning, stands out for its exceptional performance, achieving a best loss of 0.0932 and an accuracy of 69.44%. This enhancement in results across all metrics, including qualitative ones, was anticipated, given that this model represents the culmination of our developmental efforts.

Significantly, this model excels in adhering to logical conventions, such as consistently using uppercase letters for predicate names and accurately identifying each quantifier and logical connector. Notably, it successfully addresses the previously challenging 'either-or' expressions, demonstrating a comprehensive understanding of logical operators that had eluded earlier models. Moreover, it showcases the most adept selection of predicate arity, indicating a refined approach to translating natural language into first-order logic.

Despite these advancements, there remains room for improvement in predicate selection. For instance, $StudiesMathematics(x)$, while not incorrect, could be more elegantly expressed as $Studies(x, \text{mathematics})$ to better reflect the relational nature of studying a subject. This example highlights an area where further fine-tuning could enhance the model's translation fidelity.

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong." Albert Einstein.

Overall, Experiment 35's achievements underscore the effectiveness of curriculum learning in enhancing model performance, particularly in mastering the nuances of logical expression translation from natural language. See full translations below

1. $\forall x\,(\text{Student}(x) \rightarrow (\text{Studies}(x, \text{math}) \vee \text{Studies}(x, \text{physics})))$
2. $\forall x\,(\text{Student}(x) \wedge \text{StudiesMathematics}(x) \wedge \text{AttendsLogicClass}(x) \rightarrow \text{PassesLogicExam}(x))$
3. $\forall x\,(\text{MathematicsProfessor}(x) \rightarrow \text{Researcher}(x))$
4. $\forall x\,(\text{Study}(x, \text{computer science}) \leftrightarrow \text{Study}(x, \text{biology}))$
5. $\neg\text{Study}(\text{jaime}, \text{chemistry})$
6. $\exists x\,(\text{Professor}(x) \wedge \text{TeachesMathematics}(x) \wedge \text{TeachesPhysics}(x))$
7. $\exists x\,(\text{WonCompetition}(x, \text{oscar}) \wedge \neg \text{WonCompetition}(x, \text{oscar}))$
8. $\forall x\,(\text{TeachesPhysics}(x) \rightarrow \text{Genius}(x))$
9. $\forall x\,(\text{PassesLogicExam}(x) \rightarrow \text{Awarded}(x, \text{medal}))$
10. $\exists x\,(\text{Genius}(x) \wedge \text{Researcher}(x))$

Experiments 40 and 49

Experiments 40 and 49 extend the methodology of Experiment 35 by implementing quantization at 4-bit and 8-bit levels. Experiment 40 achieved a best loss of 0.0965 and an accuracy of 61.11%, whereas Experiment 49 recorded a slightly better loss of 0.0949 and an accuracy of 63.88%. These results indicate the impact of quantization on model efficiency and performance.

The translation quality of the premises for qualitative analysis in both experiments largely mirrors that of Experiment 35, with minor deviations. A notable observation is the slight decline in the precision of arity selection, as evidenced by predicates such as $WonMathematicCompetition(Oscar)$ produced by the 49th model. Despite these small discrepancies, the core logical structures and conventions are maintained.

However, a significant issue emerged in Experiment 40, where the model failed to accurately translate the 7th premise, *"Oscar is a student who has won a Mathematics competition or a Physics competition but not both"*. This omission underscores the challenges quantization may pose to maintaining translation quality, particularly in handling complex logical expressions.

In summary, while Experiments 40 and 49 demonstrate the feasibility of model quantization with minimal impact on overall translation accuracy and logical fidelity, they

"Observation is a passive science, experimentation an active science." Claude Bernard.

also highlight specific areas, such as arity management and complex premise translation, where performance can be affected. Notably, among the experiments, only Experiments 35 and 40 were able to reason correctly that the conclusion is true, underscoring their superior understanding and application of logical principles. This capability to accurately deduce the truth value of conclusions from a set of premises not only illustrates the advanced logical reasoning potential of these models but also points to the effectiveness of the fine-tuning and optimization strategies employed. Such insights emphasize the critical role of model configuration, including the application of curriculum learning and quantization, in enhancing the model's ability to navigate complex logical constructs. Find first the results for 40 and then for 49 experiments

Results for 40:

1. $\forall x\,(\text{Student}(x) \rightarrow (\text{StudiesMathematics}(x) \vee \text{StudiesPhysics}(x)))$
2. $\forall x\,(\text{Student}(x) \wedge \text{StudiesMathematics}(x) \wedge \text{AttendsLogicClass}(x) \rightarrow \text{PassesLogicExam}(x))$
3. $\forall x\,(\text{MathematicsProfessor}(x) \rightarrow \text{Researcher}(x))$
4. $\text{StudiesComputerscience}(x) \leftrightarrow \text{StudiesBiology}(x)$
5. $\neg\text{Study}(\text{jaime}, \text{chemistry})$
6. $\exists x\,(\text{Professor}(x) \wedge \text{TeachesMathematics}(x) \wedge \text{TeachesPhysics}(x))$
7. $\forall x\,(\text{WonCompetition}(x, y) \wedge (\text{Mathematics}(x) \vee \text{Physics}(x)) \rightarrow \neg\text{WonBoth}(x))$
8. $\forall x\,(\text{TeachesPhysics}(x) \rightarrow \text{Genius}(x))$
9. $\forall x\,(\text{StudiesAnything}(x) \rightarrow \text{Student}(x))$
10. $\forall x\,(\text{PassesLogicExam}(x) \rightarrow \text{Award}(x, \text{medal}))$
11. $\exists x\,(\text{Genius}(x) \wedge \text{Researcher}(x))$

results for 49:

1. $\forall x\,(\text{Student}(x) \rightarrow (\text{Studies}(x, \text{mathematics}) \vee \text{Studies}(x, \text{physics})))$
2. $\forall x\,(\text{Student}(x) \wedge \text{StudiesMathematics}(x) \wedge \text{AttendsLogicClass}(x) \rightarrow \text{PassesLogicExam}(x))$
3. $\forall x\,(\text{MathematicsProfessor}(x) \rightarrow \text{Researcher}(x))$
4. $\forall x\,(\text{Study}(x, \text{computer science}) \leftrightarrow \text{Study}(x, \text{biology}))$
5. $\neg\text{Study}(\text{jaime}, \text{chemistry})$

"In mathematics, you don't understand things. You just get used to them."John von Neumann.

6. $\exists x\,(\text{Professor}(x) \wedge \text{Teaches}(x, \text{mathematics}) \wedge \text{Teaches}(x, \text{physics}))$

7. $(\text{WonMathematicsCompetition}(\text{oscar}) \vee \text{WonPhysicsCompetition}(\text{oscar})) \wedge \neg\,(\text{WonMathematicsCompetition}(\text{oscar}) \wedge \text{WonPhysicsCompetition}(\text{oscar}))$

8. $\forall x\,(\text{Teach}(x) \rightarrow \text{Genius}(x))$

9. $\forall x\,(\text{Study}(x) \rightarrow \text{Student}(x))$

10. $\forall x\,(\text{PassesLogicExam}(x) \rightarrow \text{AwardedMedal}(x))$

11. $\exists x\,(\text{Genius}(x) \wedge \text{Researcher}(x))$

5.3 Comparison of Results

Following the detailed exploration and analysis of the experiments conducted, a comprehensive comparison of the outcomes is presented in Table 5.1. This table succinctly encapsulates the performance metrics across different models and experimental conditions, ranging from the base model used, whether quantization was applied, the dataset composition, to the presence of bias in evaluation, alongside the achieved loss and accuracy percentages.

It is pertinent to mention that 'biased' indicates that the model was exposed to some or all of the test data during the training phase. This exposure potentially influences the model's performance metrics, particularly regarding accuracy, as the model may recognize patterns from the training set.

	Base Model	**Qua**	**Dataset**	**Biased**	**Loss**	**Acc**
21	LLaMA-2-7b	none	FOLIO (6k)+ MALLS (10k)	Yes	0.1422	67.59
22	Zephyr-7b	4-bit	FOLIO (6k)	Yes	0.1068	56.48
27	Zephyr-7b	4-bit	MALLS (28k)	No	0.1063	53.70
34	LLaMA-2-7b	none	Combined	No	0.2091	67.59
35	LLaMA-2-7b	none	Combined	No	0.0932	69.44
40	LLaMA-2-7b	4-bit	Combined	No	0.0965	61.11
49	LLaMA-2-7b	8-bit	Combined	No	0.0949	63.88

Table 5.1: Comparison of Model Experiments

Furthermore, Table 5.2 extends the analysis by comparing the performance of different models, including variations of Llama-2-7b with and without quantization, against established models like GPT-3, Codex, and several versions of BERT and RoBERTa fine-tuned in (Han et al., 2022b). These comparisons highlight the significant advancements achieved through the fine-tuning and optimization processes detailed in this thesis. It's important to note that the results between my fine-tuned models and the others are not

"In mathematics, the art of asking questions is more valuable than solving problems." Georg Cantor.

directly comparable due to differences in the test datasets. The dataset used for my models is described in this chapter, while the dataset for the other models, although from FOLIO, corresponds to a new and more complete version, as the related article (Han et al., 2022b) is still in the process of being updated.

Model	**Syntactic Validity**	**Fuzzy Match**	**Accuracy**
GPT-3 davinci	94.55	58.92	46.87
Codex davinci	96.39	62.42	54.89
LLaMA-2-7b	**100**	**85.49**	**69.44**
LLaMA-2-7b q8	100	85.19	63.88
LLaMA-2-7b q4	99	84.73	61.11
BERT-base 110M	-	-	56.83
BERT-large 340M	-	-	59.03
RoBERTa-base 110M	-	-	56.83
RoBERTa-large 340M	-	-	62.11

Table 5.2: Performance comparison of various models on different metrics.

Final Remarks

We have taken a deep dive into the performance and capabilities of models fine-tuned for translating natural language into first-order logic. This evaluation involved a series of experiments, each designed to probe different aspects of model functionality, from syntactic validity, fuzzy match in the predicates and final truth value.

We began by establishing a robust set of metrics to guide our analysis, ensuring a thorough examination of both quantitative and qualitative facets of model performance. These metrics provided an understanding of the models' strengths and limitations, highlighting areas for improvement and innovation.

Our exploration of key experiments revealed significant findings. We observed that models trained with curriculum learning consistently outperformed those without, underscoring the value of a structured learning approach. Additionally, the experiments demonstrated the impact of dataset composition and quantization techniques on model outcomes, offering valuable insights into the optimization of large language models for specific tasks.

The comparative analysis against established models like GPT-3 and Codex further contextualized our findings, showcasing the advancements achieved through meticulous fine-tuning and optimization. While the performance metrics varied across models, our results highlighted the potential of Llama-2-7b, particularly when fine-tuned with a carefully curated dataset and curriculum learning strategy.

We'll finalize the journey of this research in the next chapter gathering insights and drawing conclusions as well as dicussing future alternatives of research.

"Mathematics is the queen of the sciences and number theory is the queen of mathematics." Carl Friedrich Gauss.

Chapter 6

Conclusions and Future Work

This thesis embarked on a journey to develop and validate an innovative approach for translating natural language into first-order logic using the 7 billion parameter version of Llama2, fine-tuned with Ludwig. The research aimed to strike a balance between model size and computational efficiency, employing advanced techniques like LoRa and quantization to enhance performance while keeping computational demands in check. A pivotal aspect of this work was to forge a novel pathway that marries deterministic approaches, such as those exemplified by Z3, with the generative power of machine learning models, thereby bridging the gap between the structured SMT-Lib format and the fluid, unstructured nature of NL.

6.1 Achievements in Line with Objectives

Understanding of Large Language Models

Through extensive exploration and experimentation, this thesis deepened the understanding of LLMs, shedding light on their functionality and potential for complex NLP tasks. An in-depth exploration of the general functioning of Llama and LLMs was undertaken in Chapter 1, covering everything from the basics of neural networks to the workings of generative models like GPT and Llama itself. This foundational knowledge laid the groundwork for demonstrating how LLMs, particularly Llama2, can be effectively harnessed for the nuanced task of translating natural language into first-order logic.

Comprehension of the NL to FOL Translation Process

A comprehensive understanding of FOL and the translation process from NL to FOL was achieved, significantly enriching the thesis's exploration into this complex domain. In Chapter 2, we delved into the intricate workings of FOL, from its syntax to the satisfiability of a set of formulas, and looked into the challenging problems of translating FOL into NL and, more complex still, NL into FOL. The detailed examination in Chapter 2 laid a solid foundation for the subsequent development and validation of the translation model, enabling a deep dive into the nuances of translating natural language into a structured logical format that bridges human and machine understanding.

"Lord, protect me from my friends; I can take care of my enemies." Robert Greene.

Building and Utilizing a Specialized Dataset

A specialized dataset was designed and constructed, specifically tailored to support the model's learning process for efficient NL to FOL translation. This dataset played a crucial role in refining the model's performance and its ability to handle complex logical constructs.

Fine-Tuning of Llama2

The custom dataset facilitated the fine-tuning of Llama2, with various configurations explored through Ludwig to optimize the model's translation performance. Experiments conducted underscored the effectiveness of this approach in enhancing the model's accuracy and efficiency.

Development of FOL to SMT-Lib Parser and Reasoning Process

Creating a parser for converting FOL into SMT-Lib format and establishing a reasoning process using Z3 marked a significant achievement. This development enabled the translation of NL premises and conclusions into validity statements, showcasing the model's proficiency in logical reasoning.

Evaluation of the Translation Process

The performance of the translation process was rigorously assessed, utilizing metrics that confirmed the precision and accuracy of the NL to FOL translations. The experiments provided empirical evidence of the model's capabilities and the impact of different configurations during fine-tuning as well as different training datasets.

Identification of Improvement Opportunities

Finally, the analysis identified potential gaps and outlined areas for future research and improvements. Despite the progress made, there remains scope for enhancing the model's understanding of logical operators and refining its handling of complex expressions.

6.2 Future Work

1. **Building a Larger Dataset**: The dataset's size and quality are paramount, especially in fine-tuning a model like Llama 2, which already exhibits proficiency in language comprehension. The current fine-tuning process utilized 13,278 samples, a modest number given the model's capacity. Inspired by works like Alpha Geometry (Trinh, Wu, Le, He, & Luong, 2024), which employs millions of samples, the goal is to expand our dataset significantly. This could involve generating an accurate artificial dataset or an extensive expert-created one.

2. **Exploring Alternative LLMs**: Although Llama2 was the latest and most advanced open-source LLM available at the onset of this thesis, newer models like

"AI will probably most likely lead to the end of the world, but in the meantime, there'll be great companies." Sam Altman.

Llemma (Azerbayev et al., 2023), and the Falcon LLM series present exciting alternatives to consider for future experiments.

3. **Training a Reasoning Model**: Instead of a translation-focused approach, training a model that can process an entire set of premises for reasoning could enhance consistency and accuracy, eliminating the need for multiple calls and adjustments for predicate names.

4. **Implementing a Reward System**: Introducing a reward system that reinforces correct FOL syntax generation and logical reasoning could further refine Llama's learning process and improve overall performance.

The horizons of AI and machine learning are ever-expanding, and the work presented in this thesis is but a bedrock toward more sophisticated and nuanced language models. The future is ripe with potential for advancements that will continue to push the boundaries of what these models can achieve.

"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning." Winston Churchillr.

Appendix A

Ludwig

A.1 Introduction to Ludwig

Ludwig, a declarative machine learning framework, plays a crucial role in the configuration of our model. As an open-source tool hosted by the Linux Foundation AI & Data, Ludwig provides a user-friendly, data-driven configuration system ideal for various AI tasks, including tabular data, natural language processing, and computer vision models. Its simplicity enables rapid deployment of state-of-the-art models into production.

A.1.1 Ludwig's Configuration System

Ludwig's configuration system is centered around YAML configs, requiring at least one input and one output feature. For instance, a simple configuration in Ludwig might look like:

```
input_features:
  - name: instruction
    type: text
output_features:
  - name: output
    type: text
```

This basic configuration instructs Ludwig to use the 'instruction' column as the input feature and the 'output' column as the output feature.

A.1.2 Adapting Ludwig for LLMs

With the introduction of Ludwig version 0.8, support for LLMs was added. This involved new parameters like *model_type* and *base_model* in the configuration:

```
model_type: llm
base_model: meta-llama/Llama-2-7b-hf
input_features:
  - name: instruction
    type: text
output_features:
  - name: output
    type: text
```

(Molino & Addair, 2023) In our work, we leverage LudwigModel API for using Ludwig, requiring a YAML config that defines the training pipeline. The LudwigModel object offers various methods for preprocessing, training, evaluating, and predicting.

A.2 Ludwig's Architecture

Ludwig's architecture is built on a modular and flexible design that accommodates a wide range of machine learning tasks and data types. Its core components include:

- **Data Preprocessing:** Automatic data type detection and preprocessing, ensuring that data is in the correct format for training.
- **Model Definition:** A declarative YAML configuration system that defines the model, including input features, output features, and training parameters.
- **Training and Evaluation:** Seamless model training and evaluation, with support for various machine learning algorithms and techniques.
- **Prediction:** Easy deployment of models for prediction, enabling rapid prototyping and iteration.

This architecture supports the integration of new types of data, models, and preprocessing techniques, making Ludwig highly extensible and adaptable to future requirements.

A.3 Ludwig Ecosystem

The Ludwig ecosystem encompasses a supportive community, comprehensive documentation, and integrations with popular machine learning frameworks:

- **Community Support:** A vibrant community of users and contributors that provide support through forums, GitHub issues, and social media.
- **Documentation:** Extensive documentation covering installation, configuration, and best practices for using Ludwig effectively.
- **Integrations:** Compatibility with TensorFlow, PyTorch, and other machine learning libraries, enhancing Ludwig's flexibility and functionality.

These elements contribute to a user-friendly environment that facilitates the development of high-quality machine learning models.

A.4 Benefits of Using Ludwig

Ludwig offers several advantages for machine learning projects, including:

- **Simplicity and Accessibility:** Its declarative syntax and abstraction layer make machine learning more accessible, reducing the barrier to entry.

- **Flexibility:** Supports a wide array of data types and machine learning tasks, allowing for versatile model development.
- **Rapid Prototyping:** Enables quick experimentation and iteration, speeding up the development process.
- **Scalability:** Designed to handle both small and large datasets efficiently, facilitating the scaling of projects.

A.4.1 Potential Limitations

While Ludwig simplifies many aspects of machine learning, it has potential limitations:

- **Performance:** May experience performance bottlenecks with extremely large datasets or complex models.
- **Customization:** Highly specific or novel model architectures may require more customization than Ludwig readily supports.
- **Learning Curve:** Understanding all configuration options and effectively utilizing Ludwig's full capabilities can require a learning investment.

A.5 Fine-Tuning Configuration for FOLARIS

The fine-tuning configuration for the ultimate model we developed is defined as follows:

```
model_type: llm
base_model: meta-llama/Llama-2-7b-hf
input_features:
  - name: instructions
    type: text
output_features:
  - name: FOL
    type: text
prompt:
  template: >-
    ### instructions: {instructions}
    ### Response:
generation:
  temperature: 0.01
  max_new_tokens: 256
  top_k: 8
adapter:
  type: lora
preprocessing:
  global_max_sequence_length: 512
  split:
    type: random
    probabilities:
```

```
      - 0.9
      - 0.05
      - 0.05
trainer:
  type: finetune
  epochs: 5
  learning_rate: 0.0002
  optimizer:
    type: adamw
    weight_decay: 0.01
```

A.6 Conclusion

In the context of my thesis project, Ludwig emerged as a pivotal tool, enabling the fine-tuning of Llama2 with unparalleled ease and efficiency. The project's core objective—to translate natural language into first-order logic—demanded a robust and flexible framework capable of handling sophisticated configurations. Ludwig's declarative nature facilitated this by allowing us to fine-tune Llama2 using various configurations, including advanced techniques like quantization and LoRa adapters.

The simplicity and intuitiveness of Ludwig's training process were instrumental in our ability to rapidly experiment with a diverse array of configurations. This experimentation was crucial, as it empowered us to identify and deploy the most effective configurations for our specific task. Ludwig's flexibility in managing different configurations significantly contributed to optimizing our model's performance, showcasing its capability to harness the power of Llama2 effectively.

Moreover, the ease with which Ludwig allowed us to understand and navigate the fine-tuning process underscored its value as a tool not just for model development but also for educational purposes. By abstracting the complexities typically associated with machine learning workflows, Ludwig enabled a focus on achieving our project's goals rather than wrestling with the intricacies of model training and optimization.

In conclusion, Ludwig's impact on my thesis project was profound. It not only facilitated a streamlined and effective model fine-tuning process but also enhanced our ability to experiment and select the optimal configurations for translating natural language into first-order logic. The success of this project is a testament to Ludwig's capabilities, underscoring its potential as an indispensable tool in the landscape of machine learning and model development. (Uber Technologies, Inc., 2023) (Molino & Addair, 2023)

Appendix B

Z3 Solver

B.1 Introduction to Satisfiability Modulo Theories and Z3

Satisfiability Modulo Theories (SMT) is a decision problem that involves determining whether a logical formula is satisfiable within one or more theories. It extends beyond classical propositional logic by incorporating theories such as arithmetic, bit-vectors, arrays, and others, allowing for a richer expression of constraints and properties. The Z3 solver, developed by Microsoft Research, is a state-of-the-art tool designed to tackle this problem, offering a powerful platform for software analysis, verification, and beyond (*Quantifiers - Z3 Guide*, n.d.).

At the heart of Z3's capabilities is its adept management of propositional logic. By using predefined sorts, such as the Bool sort for Boolean expressions, Z3 can manipulate logical operations—including conjunctions, disjunctions, and implications—through a straightforward syntax. This foundational aspect enables users to construct and evaluate complex logical assertions efficiently.

Further enriching Z3's logical landscape are uninterpreted formulas and constants. These elements serve as the basic building blocks for SMT formulas, where constants are zero-argument functions. This design allows for maximal flexibility, enabling any consistent interpretation of functions and constants within the constraints of the logic. For instance, uninterpreted functions can be used to model relationships or properties whose specific computational behavior is not defined, offering a canvas for exploring a wide range of logical scenarios.

Quantifier management in Z3 introduces another layer of sophistication. While Z3 shines in the realm of quantifier-free theories, it also provides robust support for formulas involving quantifiers. Through techniques such as pattern-based quantifier instantiation and model-based quantifier instantiation, Z3 can often navigate the complexities of quantified formulas, despite the inherent challenges such formulas present.

Developed by Microsoft Research, Z3 is a forefront theorem prover adept at determining the satisfiability of logical formulas across multiple theories, making it highly suitable for tasks in software analysis and verification. It aligns well with numerous software constructs by supporting a variety of theories, thereby facilitating effective logical modeling and problem-solving.

B.2 Guide to use Z3: writing SMT-lib declarations

The primary goal of this guide is to acquaint users with Z3's usage for logical modeling and solving. Although the guide provides a foundational overview of logical modeling, it suggests consulting specialized textbooks for a deeper understanding of first-order logic and decision-making processes. It is pertinent to note that while a deep understanding of logical modeling is not essential for basic modeling tasks with Z3, it becomes crucial for developing complex models.

Z3 is designed as a foundational tool for integration within other software solutions requiring logical formula resolution. It offers a suite of API features for easy integration but does not feature standalone interfaces or direct interaction mechanisms. Its front-end language design emphasizes simplicity over linguistic complexity.

This guide utilizes the SMT-Lib format for Z3 interactions. The SMT-Lib standard, a LISP-inspired syntax, facilitates model serialization and deserialization, enhancing Z3's ability to process logical assertions and queries effectively, albeit at the expense of human readability. Z3 fully supports the primary theories defined within the SMTLIB2 standard.

Z3 scripts, extending the SMT-Lib format, execute as sequences of commands but do not constitute a programming language in the conventional sense due to the absence of constructs like loops and variables. Commands such as `'help'` and `'echo'` provide script assistance and feedback. Z3 maintains an internal stack of user formulas and declarations, with commands for declaring constants and functions. For instance, declaring a function `‘f‘` that takes an integer and a Boolean and returns an integer is demonstrated as follows:

```
(echo "starting Z3...")
(declare-const a Int)
(declare-fun f (Int Bool) Int)
```

The `‘assert‘` command introduces formulas to Z3's internal stack, with its satisfiability dependent on the existence of interpretations that validate all asserted formulas.

In the context of this thesis, Z3 was employed after transforming First-Order Logic (FOL) formulas into SMT-lib strings, as elucidated in the previous section. This process involved converting premises and the negation of the conclusion into a format interpretable by Z3 (in this case a string in SMT-Lib format) with a parser developed by the author of this thesis, and then using the `‘parse_smt2_string‘` function for direct input. The methodology focused on translating logical assertions into SMT-lib format and assessing their satisfiability through Z3.

Propositional Logic Example:

```
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(define-fun conjecture () Bool
    (=> (and (=> p q) (=> q r))
        (=> p r)))
(assert (not conjecture))
(check-sat)
```

Output:

```
unsat
```

Uninterpreted Functions and Constants Example:

```
(declare-fun f (Int) Int)
(declare-fun a () Int) ; a is a constant
(declare-const b Int) ; syntax sugar for (declare-fun b () Int)
(assert (< a 20))
(assert (< b a))
(assert (= (f 10) 1))
(check-sat)
(get-model)
```

Output:

```
sat
(
  (define-fun b () Int
    18)
  (define-fun a () Int
    19)
  (define-fun f ((x!0 Int)) Int
    1)
)
```

Example Implementation

The application of this methodology is illustrated through the conversion of specified premises and a conclusion into SMT-lib format, processed by Z3 as follows:

Given the premises and conclusion in FOL format

```
premises= ['x(Person(x) → Eat(x))','¬x (Eat(x)  fly(x))', 'Fly(Larry)',
'x Person(x)']
conclusion = '¬Eat(Larry)  x Eat(x)'
```

The way the parsers work is by converting each predicate and constant into a SMT-Lib declaration, and then the whole formula is asserted

```
(declare-fun Person (Bool) Bool)
(declare-fun Eat (Bool) Bool)
(assert (forall ((x Bool)) (=> (Person x) (Eat x))))
```

each formula is first parsed into a string that fits with SMT-Lib, it means we might declare the same function or predicate or constant many times. Z3 can handle that as long as the data type and the name are the same, in this case, we make all the declarations:

```
(declare-fun Person (Bool) Bool)
(declare-fun Eat (Bool) Bool)
(declare-fun Fly (Bool) Bool)
(assert (forall ((x Bool)) (not (and (Eat x) (Fly x)))))
(declare-const Larry Bool)
(assert (Fly Larry))
(assert (exists ((x Bool)) (Person x)))
```

Then we add the negated conclusion and call the check() function which in this case bestows unsat which means in the understanding of not contradictory premises means the negated conclusion cannot be true and then the conclusion must be true.

This example underscores Z3's capability to navigate and evaluate complex logical structures, providing a computational verification of the logical arguments presented in the thesis.

B.3 Conclusion

The utilization of Z3 Solver in our project underscored its indispensable role in bridging the gap between theoretical logic formulation and practical verification. Following the transformation of natural language into first-order logic through the fine-tuned Llama2 model, and subsequently converting FOL into SMT-Lib format (as outlined in Chapter 4), Z3 Solver emerged as the pivotal tool for evaluating the logical coherence and truth values of our premises.

Z3 Solver's capability to efficiently determine the veracity of conclusions drawn from a set of premises was instrumental in validating the effectiveness of our transformation process. By leveraging Z3, we were able to automate the reasoning process, thus significantly enhancing the reliability and speed of our logical verifications. This, in turn, contributed to a more robust understanding of the logical structures underlying natural language statements and their corresponding formal representations.

Moreover, Z3's integration into our project workflow facilitated a seamless transition from the theoretical aspects of logical inference to their practical application. The solver's precision and efficiency in handling SMT-Lib formatted statements allowed us to focus on the qualitative aspects of our research, knowing that the quantitative validation was reliably managed by Z3.

In conclusion, Z3 Solver was more than just a tool; it was a critical component of our research methodology, enabling us to achieve a high level of precision and confidence in our results. The ability of Z3 to ascertain the truth values of premises, post-transformation from natural language to SMT-Lib format, not only validated our approach but also showcased the potential for automated logical reasoning in advancing the field of computational linguistics and formal logic analysis.(*Quantifiers - Z3 Guide*, n.d.)

Bibliography

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., ... Welleck, S. (2023). Llemma: An open language model for mathematics. *arXiv*. 72

Badesa, C., Jané, I., & Jansana, R. (1998). *Elementos de lógica formal*. Ariel Filosofía. 19

Bansal, N. (2015). *Translating natural language propositions to first order logic* (Unpublished master's thesis). Indian Institute of Technology Kanpur, Kanpur, India. (A thesis submitted in Partial Fulfillment of the Requirements for the Degree of Master of Technology in the Department of Computer Science & Engineering) 32, 34, 35, 36

Barker-Plummer, D., Cox, R., & Dale, R. (2000). Dimensions of difficulty in translating natural language into first order logic. *CSLI, Stanford University, CA, USA*. (dbp@csli.stanford.edu, richc@sussex.ac.uk, rdale@science.mq.edu.au) xi

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *International conference on machine learning*. Retrieved from https://api.semanticscholar.org/CorpusID:873046 48

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., ... Weinbach, S. (2022). GPT-NeoX-20B: An Open-Source Autoregressive Language Model. *arXiv preprint arXiv:2204.06745*. (Submitted on 14 Apr 2022) 37

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*. (Submitted on 28 May 2020, last revised 22 Jul 2020) 37

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., ... Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*. (Submitted on 7 Jul 2021, last revised 14 Jul 2021) 37

Clark, P., Tafjord, O., & Richardson, K. (2020). Transformers as Soft Reasoners over Language. *arXiv preprint arXiv:2002.05867*. (Submitted on 14 Feb 2020, last revised 5 May 2020) 37

de Moura, L., & Bjørner, N. (2008). Z3: An Efficient SMT Solver. In C. R. Ramakrishnan & J. Rehof (Eds.), *Tools and algorithms for the construction and analysis of systems, tacas 2008* (Vol. 4963, pp. 337–340). Springer, Berlin, Heidelberg. doi: 10.1007/978-3-540-78800-3_24 xii

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*. (Submitted on 23 May 2023) xiii

Devianto, D., Yollanda, M., Maiyastri, M., & Yanuar, F. (2023, April 16). The soft computing ffnn method for adjusting heteroscedasticity on the time series model of currency exchange rate. *Frontiers in Applied Mathematics and Statistics*, *2023*. (Section: Mathematics of Computation and Data Science) 1

Dong, L., & Lapata, M. (2016, Jun). Language to logical form with neural attention. *In-*

stitute for Language, Cognition and Computation, School of Informatics, University of Edinburgh. xi

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Benson, L., ... Radev, D. (2022a). *Folio github repository.* https://github.com/Yale-LILY/FOLIO. 53

Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Benson, L., ... others (2022b). *Folio: Natural language reasoning with first-order logic.* ArXiv. Retrieved from https://arxiv.org/abs/2209.00840 xii, 32, 37, 40, 51, 68, 69

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*. (Submitted on 17 Jun 2021, last revised 16 Oct 2021) xiii

Kahneman, D. (2011). *Thinking, fast and slow.* Farrar, Straus and Giroux. 45

Karpathy, A. (2021). *The spelled-out intro to language modeling: building makemore.* (Available from YouTube: https://www.youtube.com/watch?v=PaCmpygFfXo) 3

Karpathy, A. (2023). *Let's build gpt: from scratch, in code, spelled out.* https://www.youtube.com/watch?v=kCc8FmEb1nY&list=PLAqhIrjkxbuWI23v9cThsA9GvCAUhRvKZ&index=7. (Accessed: 2024-02-27) 8

Korovin, K. (2023). *Automated reasoning for first-order logic: Theory, practice and challenges.* Manchester. (Part I, supported by a Royal Society University Fellowship) xii

Liu, J., Cui, L., Liu, H., Huang, D., Wang, Y., & Zhang, Y. (2020). Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*. 36

Miquel, A. (2017). Teorías y modelos. , 1-26. 26

Molino, P., & Addair, T. (2023). *Efficient fine-tuning for llama-v2-7b on a single gpu.* https://www.youtube.com/live/g68qlo9Izf0?si=ADC_m_-BLdbwx74f. (Webinar presented by DeepLearningAI, August 29) 11, 74, 76

Pelayo, A. M. (2023a). *Folaris_4bits model on hugging face.* https://huggingface.co/Fermat111/FOLARIS_4bits. 57

Pelayo, A. M. (2023b). *Fol dataset on hugging face.* https://huggingface.co/datasets/Fermat111/FOL/blob/main. 59

Pelayo, A. M. (2023c). *Google colab notebook.* https://colab.research.google.com/drive/1JOS2ZWI_ZEp1UT_8AVsE_2Qv--x-vkvO?usp=sharing. 57

Pelayo, A. M. (2023d). *Nl2fol github repository.* https://github.com/AngelMisaelPelayo/NL2FOL. 54, 57

Perikos, I., Grivokostopoulou, F., & Hatzilygeroudis, I. (2023). Teaching assistant tools for nl to fol conversion. *University of Patras, School of Engineering, Department of Computer Engineering & Informatics, 26500 Patras, Hellas (Greece).* xi

Quantifiers - z3 guide. (n.d.). https://microsoft.github.io/z3guide/docs/logic/Quantifiers. (Accessed: yyyy-mm-dd) 77, 80

Singh, H., Aggrawal, M., & Krishnamurthy, B. (2020). Exploring neural models for parsing natural language into first-order logic. *ArXiv, abs/2002.06544*. xi

Sinha, K., Sodhani, S., Dong, J., Pineau, J., & Hamilton, W. L. (2019, November). CLUTRR: A diagnostic benchmark for inductive reasoning from text. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 4506–4515). Hong Kong, China: Association for Computational Linguistics. Retrieved from https://aclanthology

.org/D19-1458/ doi: 10.18653/v1/D19-1458 37

Tang, W., & Belle, V. (2024). LTLBench: Towards Benchmarks for Evaluating Temporal Logic Reasoning in Large Language Models. *arXiv preprint arXiv:2407.01234*. (Submitted on 7 Jul 2024) 37

Tarski, A. (1999). La concepción semántica de la verdad y los fundamentos de la semántica. *A Parte Rei*. 20

Tian, J., Li, Y., Chen, W., Xiao, L., He, H., & Jin, Y. (2021, November). Diagnosing the first-order logical reasoning ability through LogicNLI. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 3738–3747). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.emnlp-main.303/ doi: 10.18653/v1/2021.emnlp-main.303 37

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*. (Submitted on 18 Jul 2023, last revised 19 Jul 2023) xi

Trinh, T. H., Wu, Y., Le, Q. V., He, H., & Luong, T. (2024). Solving olympiad geometry without human demonstrations. *Nature*, *625*(7995), 476–482. Retrieved from https://doi.org/10.1038/s41586-023-06747-5 doi: 10.1038/s41586-023-06747-5 71

Uber Technologies, Inc. (2023). *User guide.* https://ludwig.ai/latest/user_guide/. 76

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*. 4, 5, 8, 9

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., ... Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903*. (Submitted on 28 Jan 2022, last revised 10 Jan 2023) 40

Yang, Y., Xiong, S., Payani, A., Shareghi, E., & Fekri, F. (2023). *Harnessing the power of large language models for natural language to first-order logic translation.* ArXiv. Retrieved from https://arxiv.org/abs/2305.15541 xi, 12, 32, 37, 40, 41, 42, 44, 53

Yu, W., Jiang, Z., Dong, Y., & Feng, J. (2020). Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*. 36

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., ... Zettlemoyer, L. (2022). OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*. (Submitted on 2 May 2022, last revised 21 Jun 2022) 37

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... Zhang, B. (2023). A survey of large language models.
9

Ángel Misael Pelayo Gómez

Translating Thought Advancing First-Order Logic Interpretation with Llama2 and Z3 Solver.pdf

Universidad Michoacana de San Nicolás de Hidalgo

Detalles del documento

Identificador de la entrega
trn:oid:::3117:432356056

Fecha de entrega
20 feb 2025, 1:01 p.m. GMT-6

Fecha de descarga
20 feb 2025, 1:05 p.m. GMT-6

Nombre de archivo
Translating Thought Advancing First-Order Logic Interpretation with Llama2 and Z3 Solver.pdf

Tamaño de archivo
2.2 MB

100 Páginas

34,104 Palabras

189,254 Caracteres

22% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Grupos de coincidencias

- **432** Sin cita o referencia 15%
  Coincidencias sin una citación ni comillas en el texto
- **33** Faltan citas 1%
  Coincidencias que siguen siendo muy similar al material fuente
- **98** Falta referencia 6%
  Las coincidencias tienen comillas, pero no una citación correcta en el texto
- **1** Con comillas y referencia 0%
  Coincidencias de citación en el texto, pero sin comillas

Fuentes principales

- 19% Fuentes de Internet
- 15% Publicaciones
- 0% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

A quien corresponda,

Por este medio, quien abajo firma, bajo protesta de decir verdad, declara lo siguiente:

- Que presenta para revisión de originalidad el manuscrito cuyos detalles se especifican abajo.
- Que todas las fuentes consultadas para la elaboración del manuscrito están debidamente identificadas dentro del cuerpo del texto, e incluidas en la lista de referencias.
- Que, en caso de haber usado un sistema de inteligencia artificial, en cualquier etapa del desarrollo de su trabajo, lo ha especificado en la tabla que se encuentra en este documento.
- Que conoce la normativa de la Universidad Michoacana de San Nicolás de Hidalgo, en particular los Incisos IX y XII del artículo 85, y los artículos 88 y 101 del Estatuto Universitario de la UMSNH, además del transitorio tercero del Reglamento General para los Estudios de Posgrado de la UMSNH.

Datos del manuscrito que se presenta a revisión		
Programa educativo	Maestría en Ciencias en Ingeniería Eléctrica (Sistemas Computacionales)	
Título del trabajo	Translating Thought: Advancing First-Order Logic Interpretation with Llama2 and Z3 solver.	
	Nombre	**Correo electrónico**
Autor/es	Ángel Misael Pelayo Gómez	angel.pelayo@umich.mx
Director	Jaime Cerda Jacobo	jaime.cerda@umich.mx
Codirector		
Coordinador del programa	J. Avelio Medina Ríos	avelio.medina@umich.mx

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Asistencia en la redacción	Sí	Grammarly para revisar gramática y mejorar legibilidad y ChatGPT4 para refinar la estructura y el flujo del texto.

Formato de Declaración de Originalidad y Uso de Inteligencia Artificial

Coordinación General de Estudios de Posgrado
Universidad Michoacana de San Nicolás de Hidalgo

Uso de Inteligencia Artificial		
Rubro	**Uso (sí/no)**	**Descripción**
Traducción al español	No	
Traducción a otra lengua	No	
Revisión y corrección de estilo	Sí	Grammarly y ChatGPT-4 para asegurarme que el tono y cohesion fueran adecuados.
Análisis de datos	No	
Búsqueda y organización de información	Sí	ChatGPT-4 para el capitulo: "The pillars of logic: understanding First-Order Logic" (agregado en las referencias del capitulo)
Formateo de las referencias bibliográficas	Sí	Grammarly con la modalidad "Get Citation" ChatGPT-4 para pasar la referencia en latex
Generación de contenido multimedia	No	
Otro	No	

Datos del solicitante	
Nombre y firma	Ángel Misael Pelayo Gómez
Lugar y fecha	Morelia Michoacán a 14 de mayo del 2024